Открытая олимпиада школьников (информатика)
(№66 Перечня олимпиад школьников, 2024/2025 уч.год)

Содержание

Задания для 11 класса

Заключительный этап (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балла)

[100 первых цифр]

Петя составляет ряд из периодических дробей, записанных в шестнадцатеричной системе счисления. Первый член ряда равен $0,(AB)_{16}$. Каждый следующий член ряда ровно в 4 раза меньше предыдущего. Известно, что у некоторого члена ряда сумма первых 100 цифр после запятой равна 441_{10}. Найдите и укажите в ответ номер этого члена ряда. Если такого члена ряда не существует, запишите в ответ NULL.

Ответ: 117

Решение:

Обратим внимание, что деление числа на 4 равносильно сдвигу запятой на один разряд влево в записи числа в четверичной системе счисления. Также вспомним, что мы можем перевести число из шестнадцатеричной системы счисления в четверичную заменив независимо каждый шестнадцатеричный разряд на два четверичных.

Тогда первый член ряда $\mathbf{0,(AB)_{16}} = 0,(2223)_4$.

$0,(2223)_4 / 4_{10} = 0,(2223)_4 / 10_4 = 0,0(2223)_4 = \mathbf{0,2(AE)_{16}}$ – это второй член ряда.

Повторим операцию деления несколько раз, зафиксировав закономерность в записи получающихся членах ряда.

$0,0(2223)_4 / 4_{10} = 0,0(2223)_4 / 10_4 = 0,00(2223)_4 = \mathbf{0,0(AB)_{16}}$ – третий член ряда.

$0,00(2223)_4 / 4_{10} = 0,00(2223)_4 / 10_4 = 0,000(2223)_4 = \mathbf{0,02(AE)_{16}}$ – четвертый член ряда.

Легко заметить, что все нечетные члены ряда будут получаться добавление одного нуля после запятой в предыдущий нечетный член ряда, а все четные – добавлением одного нуля после запятой в предыдущий четный член ряда.

Рассмотрим возможные суммы первых 100 цифр после запятой для нечетных членов ряда. Для первого члена ряда это будет 50*10+50*11 = 1050. Для третьего члена ряда 50*10+49*11 = 1039. То есть, у нечетных членов ряда сумма может быть представлена или как X*21 или как X*21-11, где X – натуральное число. Аналогично для четных членов ряда сумма будет равна или 2+Y*24-14 = Y*24-12 или 2+Y*24. Рассмотрим число из условия – 441_{10} и заметим, что это 21*21. Следовательно, это сумма цифр для нечетного члена ряда вида X*21, где X=21. Осталось вычислить его номер. Для первого члена ряда X=50, для пятого члена ряда X=49 и т.д. Тогда номер нужного члена рядам можно вычислить как (50-X)*4+1 = (50-21)*4+1=117

2. Кодирование информации. Объем информации (2 балла)

[Полёт]

Петя создает ПО для управления дроном. Дрон перемещается в трехмерном пространстве с заданной декартовой системой координат. Дрон умеет выполнять 6 возможных команд: вверх, вниз, вправо, влево, вперед и назад. Каждая команда перемещает дрон на 1 единицу длины в указанном направлении. Пространство не содержит никаких препятствий для перемещения дрона. Программа для дрона – это последовательность из определенного количества указанных команд. Пока дрон умеет выполнять только программы, состоящие ровно из 11 команд. Петя решил записывать программы для дрона в память в виде последовательности команд, кодируя каждую команду минимально возможным, одинаковым для всех команд количеством бит.

Вася заметил, что дрон всегда стартует из точки A и всегда заканчивает движение в точке B, смещенную на 2 единицы длины вправо, 3 единицы длины вверх и 4 единицы длины вперед. Вася сказал Пете, что тогда нет смысла выделять память исходя из необходимости хранить произвольную программу заданной длины. Он предложил перенумеровать натуральными числами все программы, которые смогут переместить дрона из точки A в точку B и сохранять в памяти номер такой программы, используя для каждого возможного номера минимально возможное, одинаковое для всех номеров количество бит. Насколько меньше бит потребуется Васе для хранения одной программы, чем предлагал Петя? В ответе укажите целое число.

Ответ:16

Решение:

Определим объём информации, который необходимо выделить для программы, записывая её способом Пети. Для кодирования шести команд будет достаточно $\lceil \log_2 \mathbf{6} \rceil = \mathbf{3}$ бит информации, соответственно, для 11 команд будет достаточно $\mathbf{3 \cdot 11 = 33}$ бит.

Определим объём информации, который необходимо выделить для программы, записывая её способом Васи. Для удобства обозначим команды буквами: вверх - U, вниз - D, вправо - R, влево - L, вперед - F, назад - B. Запишем минимальную программу, которая нужна для смещения на 2 единицы длины вправо, 3 единицы длины вверх и 4 единицы длины вперед: RRUUUFFFF. Эта программа состоит из 9 команд, но по условию в программе должно быть 11 команд, при этом итоговое перемещение дрона в пространстве должно остаться таким же. Для этого итоговая программа должна иметь вид RRUUUFFFFXY, где X и Y - две команды, выполняющие движение в противоположном направлении (либо U и D, либо R и L, либо F и B).

Посчитаем количество различных программ, которые могут быть сведены к RRUUUFFFFUD. Необходимо посчитать количество перестановок с повторениями, для этого можно либо воспользоваться готовой формулой, либо вывести формулу из тех рассуждений, что всего всего есть $\boldsymbol{P_{11} = 11!}$ перестановок 11 элементов, при этом перестановки, в которых меняются местами одинаковые буквы, мы не учитываем. Таких перестановок будет $\mathbf{2!} \cdot \mathbf{(3+1)!} \cdot \mathbf{4!} \cdot \mathbf{1!}$: две буквы R из минимальной программы, три буквы U из минимальной программы и ещё одна из пары UD, четыре буквы F из минимальной программы и одна буква D из пары UD. Тогда формула будет выглядеть следующим образом:

$$\frac{\mathbf{11!}}{\mathbf{2!} \cdot \mathbf{(3+1)!} \cdot \mathbf{4!} \cdot \mathbf{1!}} = \mathbf{34650}$$

Итого получилось 34650 программ.

Аналогичные рассуждения можно провести для программ RRUUUFFFRL и RRUUUFFFFB. Для первого случая количество программ равно:

$$\frac{\mathbf{11}!}{(\mathbf{2}+\mathbf{1})!\cdot\mathbf{3}!\cdot\mathbf{4}!\cdot\mathbf{1}!}=\mathbf{46200}$$

Для второго случая количество программ равно:

$$\frac{\mathbf{11}!}{\mathbf{2}!\cdot\mathbf{3}!\cdot(\mathbf{4}+\mathbf{1})!\cdot\mathbf{1}!}=\mathbf{27720}$$

И тогда на хранение номера одной программы нужно будет выделить $\lceil\log_2(\mathbf{34650}+\mathbf{46200}+\mathbf{27720})\rceil=\lceil\log_2\mathbf{108570}\rceil=\mathbf{17}$ бит.

Итоговый выигрыш по сравнению со способом Пети составит $\mathbf{33}-\mathbf{17}=\mathbf{16}$ бит.

3. Основы логики (3 балла)
[Кто такой Жегалкин?]

Есть логический преобразователь, на вход которому подается логическая функция от 20 переменных, не являющаяся тождественной ложью (то есть имеющая значение ИСТИНА хотя бы для одного набора значений переменных). Работает преобразователь следующим образом:

1. Строится таблица истинности данной функции.
2. Из полученной таблицы истинности берутся наборы переменных, для которых значение функции равно ИСТИНЕ.
3. Для каждого такого набора составляется полная конъюнкция всех 20 переменных, причем если значение этой переменной в наборе равно ЛЖИ, то в конъюнкции эта переменная будет с отрицанием, если значение переменной равно ИСТИНЕ, то в конъюнкции эта переменная будет без изменений.
4. Между всеми полученными полными конъюнкциями ставятся операции $\oplus$ (Исключающее ИЛИ).
5. Все переменные с отрицанием заменяются по следующему правилу:
   $\bar{X}=(X\oplus 1)$
6. Раскрываются все скобки по правилу: $X\wedge(Y\oplus Z)=X\wedge Y\oplus X\wedge Z$. После раскрытия скобок должно получиться выражение, которое может содержать некоторые отдельные переменные, конъюнкции переменных и константу ИСТИНА, соединенные Исключающим ИЛИ.
7. Из выражения убираются повторяющиеся конъюнкции. В результате у нас получается новая форма записи исходной функции. Гарантируется, что для каждой входной функции эта форма уникальна.
8. Производится проверка, остались ли в получившейся форме записи выражения операции конъюнкции. Если остались – преобразователь возвращает ЛОЖЬ, если нет – возвращает ИСТИНА.

Сколько существует таких функций от 20 переменных, которые можно подать на вход преобразователю и для которых он вернет ИСТИНУ? В ответе укажите целое число.

Ответ: 2097151

Решение:

Разберем работу алгоритма. Выполнив пункты 1-3 мы получаем набор полных конъюнкций, которые при сложении дают СДНФ исходной функции (то есть дизъюнкцию конъюнкций, состоящих из полного набора переменных, от которых зависит исходная функция). Только на шаге 4 вместо ожидаемых дизъюнкций полные конъюнкции объединяются операцией Исключающее ИЛИ. Следовательно получается запись, вида A $\oplus$ B $\oplus$ C…, где A, B, C…. – полные конъюнкции, в которых могут присутствовать инвертированные переменные.

Проанализируем, что будет, если все инвертированные переменные заменяются на (a $\oplus$ 1) и по указанным правилам раскрываются скобки. Если в конъюнкции присутствовала одна инвертированная переменная, то X(a $\oplus$ 1) = Xa $\oplus$ X. Если в конъюнкции присутствовали две инвертированные переменные, то
X(a $\oplus$ 1)(b $\oplus$ 1)= (Xa $\oplus$ X)(b $\oplus$ 1) = (Xa $\oplus$ X)b $\oplus$ 1(Xa $\oplus$ X) = Xab $\oplus$ Xb $\oplus$ Xa $\oplus$ X

И так далее, если посмотреть, можно заметить, что после раскрытия скобок мы получаем форму записи вида A $\oplus$ B $\oplus$ C $\oplus$ …, где A, B, C, … – конъюнкции, в которых уже не обязательно присутствуют все переменные, так же важно отметить, что в них не будет инвертированных переменных, и эта конъюнкция может быть из одной переменной (либо какая то переменная, от которой зависит исходная функция, либо константа 1). Так же после избавления от пар повторяющихся конъюнкций, понимаем, что A, B, C… будут уникальными конъюнкциями.

Посчитаем, сколько у нас может быть функций от 20 переменных. Помним, что тождественная ложь нам не подходит по условию задачи. Тогда функций получается: $2^{(2\wedge 20)}-1$

Посчитаем, сколько у нас всего может получиться форм записи, полученных после 7-го пункта алгоритма. В каждую конъюнкцию каждая переменная у нас может как входить, так и не входить (если в конъюнкцию не входит ни одна переменная – это соответствует случаю, когда конъюнкция представляет из себя константу 1). Следовательно, всего вариантов возможных конъюнкций 2^{20}. Далее каждая такая конъюнкция может как входить, так и не входить в конечную форму записи, при этом хотя бы одна конъюнкция в конечной форме должна быть => всего таких форм у нас $2^{(2\wedge 20)}-1$.

По условию задачи, мы знаем, что для каждой входной функции получается своя уникальная форма записи. Так же мы получили, что количество входных функций и количество возможных форм записи одинаковы, следовательно, у каждой входной функции есть единственная и уникальная форма записи, представленная в алгоритме. Тогда, нам достаточно посчитать те формы, в которых конъюнкции представляют из себя одну переменную: всего таких конъюнкций будет 20+1=21 (20 переменных от которых зависит исходная функция и константа 1). Каждую такую конъюнкцию мы можем как включать, так и не включать в запись, но хотя бы одна конъюнкция должна присутствовать => всего удовлетворяющих нас форм записи (а значит и функций) = $2^{(n+1)}-1=2^{21}-1$.

4. Кодирование информации. Алгоритмы обработки кодированной информации (1 балл) [Разбегайся!]

Пусть N – шестизначное десятичное число. Последовательность цифр R получается по следующему алгоритму:

1. 1000 раз повторяется первая (самая левая) цифра числа N.
2. Цикл для всех последующих цифр числа N, двигаясь слева направо:

   После каждого элемента последовательности, полученной на предыдущем шаге цикла вставить два раза подряд эту цифру

Например, если N=123456, то первые 10 элементов последовательности R, получившейся после завершения алгоритма будут:

1, 6, 6, 5, 6, 6, 5, 6, 6, 4

Известно, что в получившейся последовательности элемент с номером 209161 равен 7, элемент с номером 242758 равен 3, а элемент с номером 189890 равен 1. Нумерация элементов слева направо с 1. Определите максимальное число N, при котором это возможно, и запишите его в ответ. Если такое число невозможно, запишите в ответ NULL.

Ответ: 399791

Решение:

Пусть исходное число: 123456, то есть его цифры соответствуют их номерам считая слева направо от 1.

Рассмотрим начало последовательности после каждого шага цикла:

122122122122…

133233233133…

Обратим внимание, что после каждого шага последовательность будет состоять из одинаковых подпоследовательностей, начинающихся с каждой из тысячи единиц. Также легко заметить, что длина такой подпоследовательности будет утраиваться на каждом шаге, а, значит, после вставки последней цифры (6) длина каждой подпоследовательности, начинающейся с 1 будет 3^5=243.

Рассмотрим элемент результирующей последовательности с номером 209161. Поделим номер на 243 с остатком. Остаток будет равен 181. То есть, это 181-й элемент подпоследовательности, начинающейся с 1. Обратим внимание, что в такой подпоследовательности, полученной на определенном шаге цикла, на всех позициях, остаток от деления номера которых на 3 равен 0 или 2 стоят цифры, которые вставлены на этом шаге. Но остаток от деления 181 на 3 равен 1. Значит, это не может быть последняя цифра исходного числа.

При переходе на следующий шаг цикла, номер элемента, который был в последовательности на предыдущем шаге (n), будет давать новый номер n*3-2. Поэтому символ на 181 позиции в предыдущей подпоследовательности стоял на позиции (181+2)/3=61. Остаток от деления 61 на 3 равен опять 1. Значит это не цифра, стоящая на пятой позиции в исходном числе. Символ на 61 позиции в предыдущей подпоследовательности стоял на позиции (61+2)/3=21. 21 нацело делится на 3. Следовательно, мы знаем, что это цифра, которая в исходном числе стоит на позиции 4. Таким образом, мы определили, что в исходном числе на позиции 4 стоит цифра 7.

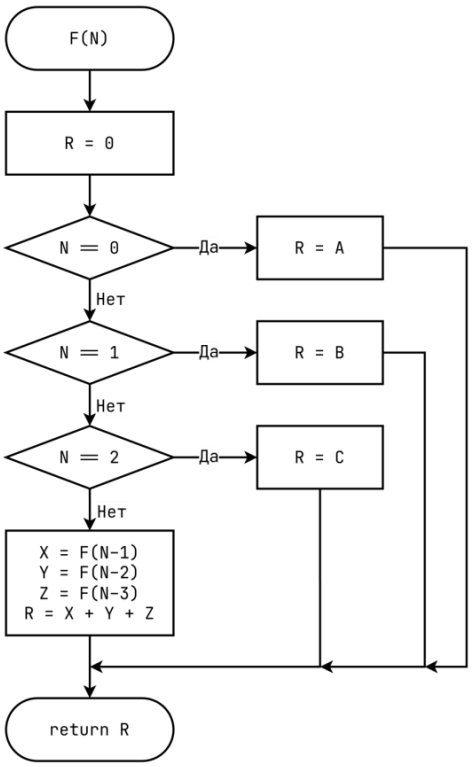
Аналогичными рассуждениями мы получим, что на позиции 1 в исходном числе стоит цифра 3 (остаток от деления 242758 на 243 равен 1), а на позиции 6 стоит цифра 1 (остаток от деления 189890 на 243 равен 107, а остаток от деления 107 на 3 равен 2). Поскольку нас просят найти максимальное число, то остальные его цифры должны быть максимальными, то есть равны 9.

Значит, это число 399791.

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (1 балл)

[Непизанские кролики]

Дана блок-схема алгоритма, реализованного в виде рекурсивно вызываемой функции:



Известно, что A, B и C - натуральные числа и что $A < B < C$.

Петя вызывает эту функцию, передавая ей в качестве входного параметра натуральное число. Известно, что для некоторого k функция возвращает следующие значения:

$F(k) = 50890368413$;

$F(k+1) = 93601980590$;

$F(k+2) = 172160883161$.

Определите значения A, B и C, при которых это возможно. Если таких вариантов несколько, выберите вариант с наименьшей суммой A, B и C. В ответе введите в указанном порядке значения A, B и C, разделённые пробелом.

Ответ: 1 2 5

Решение:

На блок-схеме изображён модифицированный алгоритм для генерации N-го числа Фибоначчи: в нём даются первые три числа, а следующие числа определяются как сумма трёх предыдущих (при A=0, B=0 и C=1 такая последовательность называется числами трибоначчи). Соответственно, нужно найти такие три числа, которые сгенерируют ряд с указанными в условии элементами. Для этого можно написать программу, которая принимает числа из условия на вход и вычитает первые два из третьего, тем самым генерируя предыдущие элементы последовательности:

```
d, e, f = 50890368413, 93601980590, 172160883161

while d > 0:
    t = f - d - e
    print(t)
    d, e, f = t, d, e
```

Результат работы программы нужно изучить достаточно внимательно, ведь в конце есть лишние элементы, которые не входят в последовательность, поскольку должно соблюдаться, что A,B,C∈N и A<B<C. В данном варианте окончание вывода будет таким:

...
15
8
5
2
1
2
-1

Две последние строки нас не интересуют, и в итоге A=1,B=2,C=5.

6. Телекоммуникационные технологии (2 балла). [BGP]

Интернет состоит из отдельных крупных сетей, управляемых отдельными организациями. Между такими сетями распределяется пространство IP-адресов и устанавливается техническое и организационное взаимодействие. Несколько

упрощая, можно сказать, что такие крупные сети называются автономными системами (AS). За каждой AS закрепляется IP-подсеть.

Для обмена маршрутной информацией между маршрутизаторами AS используется внешний протокол динамической маршрутизации BGP (Border Gateway Protocol). BGP выбирает наилучший маршрут для передачи пакетов на основе ряда атрибутов и метрик.

BGP-роутер при маршрутизации пакета выбирает IP-адрес следующего по маршруту маршрутизатора, который в BGP называется «соседом» (neighbor), по набору атрибутов маршрутов. В задаче для упрощения будем считать, что реализуется следующий алгоритм.

1. по адресу назначения в IP-пакете выбираются подходящие маршруты из таблицы маршрутизации по полю «сеть назначения».
2. потом из них выбирается лучший, для чего **последовательно** проверяются атрибуты маршрутов. То есть сначала проверяется первый атрибут всех подходящих маршрутов, и если у возможных маршрутов он имеет равное значение, проверяется второй атрибут и т.д.

В задаче используются следующие атрибуты в указанном порядке:

1. Локальное предпочтение (Local Preference)

Local Preference — это атрибут, который используется для выбора исходящего трафика из автономной системы. Маршрут с **наибольшим** значением Local Preference считается предпочтительным.

2. Наименьший Путь AS (AS Path)

BGP предпочитает маршруты с **наименьшим** количеством автономных систем в пути (AS Path). Чем короче путь, тем лучше.

3. Наименьший IGP Metric до Next Hop

Если есть несколько маршрутов с одинаковыми атрибутами, BGP выбирает маршрут с **наименьшим** значением метрики IGP до Next Hop (следующего прыжка).

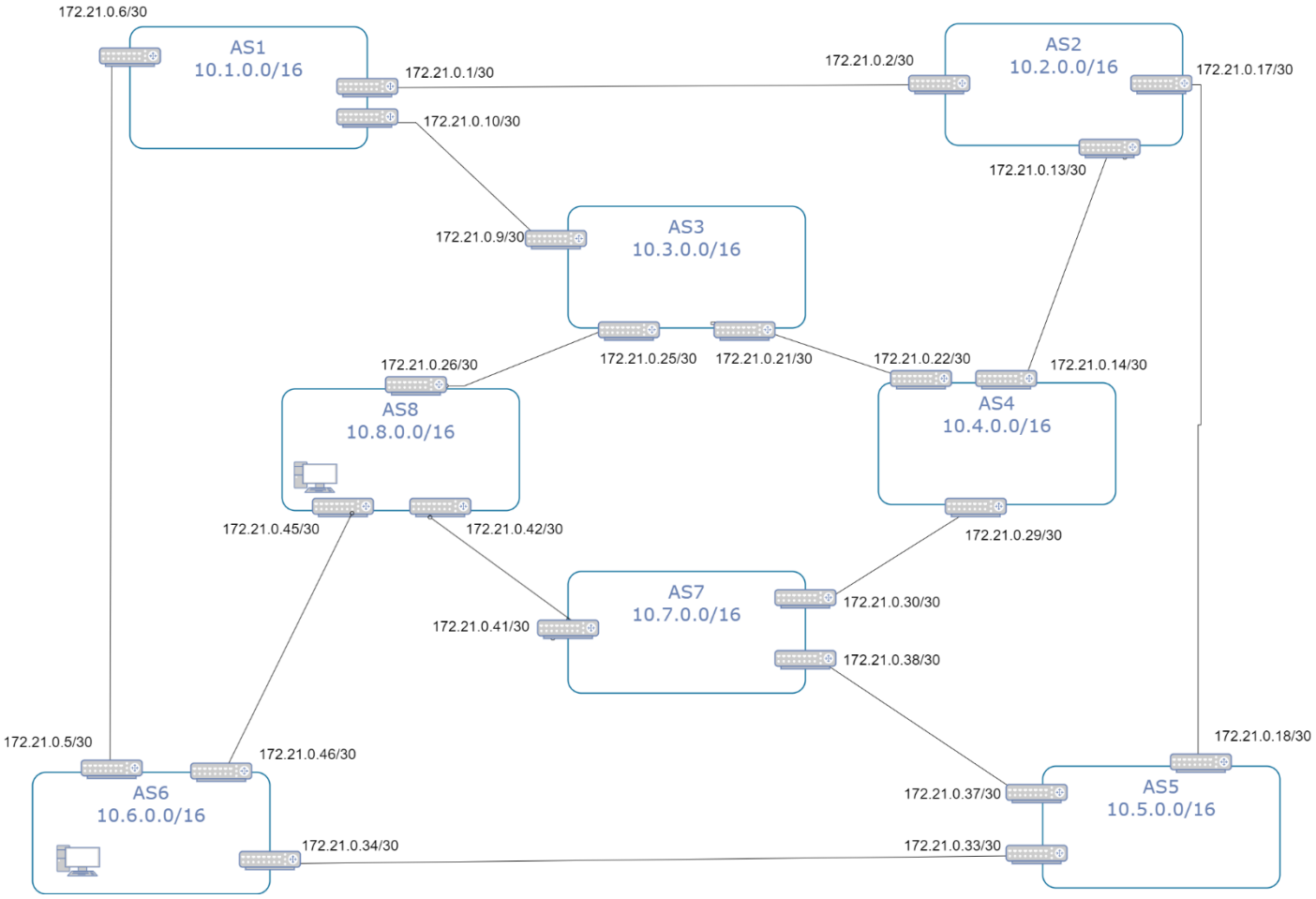
4. Старейший маршрут (Oldest Route)

Если все предыдущие атрибуты равны, BGP предпочитает **старейший** маршрут, так как он считается более стабильным.

5. Наименьший IP-адрес соседа

В крайнем случае, если все предыдущие атрибуты равны, BGP выбирает маршрут, полученный от соседа с **наименьшим** IP-адресом (сравнение производится как сравнение 32-битных значений адресов).

Далее приведены схема сети и фрагменты таблиц маршрутов для каждой из AS (сделано это для простоты, в реальности эти данные есть в каждом граничном маршрутизаторе AS).



AS1:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.2
10.1.0.0/16	Локально	500	0	0	2025-02-01	-
10.3.0.0/16	AS3	200	1	10	2025-02-03	172.21.0.9
10.7.0.0/16	AS2	100	3	20	2025-02-03	172.21.0.2
10.8.0.0/16	AS3	100	2	20	2025-02-01	172.21.0.9
10.6.0.0/16	AS6	100	2	20	2025-02-01	172.21.0.5
10.6.0.0/16	AS3	100	2	20	2025-02-01	172.21.0.9
10.3.0.0/16	AS6	200	3	50	2025-02-23	172.21.0.5
10.4.0.0/16	AS3	200	2	5	2025-02-03	172.21.0.9

AS2:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.2.0.0/16	Локально	500	0	0	2025-02-01	-
10.3.0.0/16	AS1	100	2	20	2025-02-03	172.21.0.1
10.6.0.0/16	AS1	100	2	20	2025-02-10	172.21.0.1
10.7.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.14
10.8.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.6.0.0/16	AS5	100	2	20	2025-02-03	172.21.0.18
10.3.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.4.0.0/16	AS1	200	2	20	2025-02-03	172.21.0.1
10.4.0.0/16	AS4	100	1	20	2025-02-03	172.21.0.14
10.1.0.0/16	AS1	100	1	20	2025-02-03	172.21.0.1
10.1.0.0/16	AS4	200	3	20	2025-02-03	172.21.0.14

AS3:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.22
10.6.0.0/16	AS1	100	2	20	2025-02-01	172.21.0.10
10.6.0.0/16	AS4	200	4	10	2025-02-03	172.21.0.22
10.7.0.0/16	AS4	100	3	10	2025-02-04	172.21.0.22
10.3.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS8	100	3	10	2025-02-04	172.21.0.26
10.4.0.0/16	AS4	200	1	10	2025-02-04	172.21.0.22
10.4.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS1	100	1	20	2025-02-04	172.21.0.10

AS4:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.13
10.6.0.0/16	AS2	100	3	5	2025-02-03	172.21.0.13
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	AS7	100	3	10	2025-02-01	172.21.0.30
10.8.0.0/16	AS3	100	1	10	2025-02-01	172.21.0.21
10.6.0.0/16	AS3	100	3	5	2025-02-11	172.21.0.21
10.1.0.0/16	AS3	100	3	5	2025-02-01	172.21.0.21

AS5:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.4.0.0/16	AS2	100	2	10	2025-02-02	172.21.0.17
10.5.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	AS6	100	1	20	2025-02-03	172.21.0.34

10.8.0.0/16	AS7	100	3	20	2025-02-01	172.21.0.38
10.3.0.0/16	AS2	500	3	20	2025-02-01	172.21.0.17
10.2.0.0/16	AS2	100	3	20	2025-02-01	172.21.0.17
10.3.0.0/16	AS6	100	3	20	2025-02-01	172.21.0.34
10.4.0.0/16	AS7	100	2	20	2025-02-02	172.21.0.38
10.1.0.0/16	AS6	100	2	20	2025-02-02	172.21.0.34
10.1.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.17

AS6:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS1	100	3	20	2025-02-02	172.21.0.6
10.6.0.0/16	Локально	500	0	0	2025-02-01	-
10.4.0.0/16	AS5	100	3	51	2025-02-11	172.21.0.33
10.7.0.0/16	AS1	100	4	20	2025-02-03	172.21.0.6
10.8.0.0/16	AS1	100	3	20	2025-02-01	172.21.0.6
10.3.0.0/16	AS8	200	2	5	2025-03-08	172.21.0.45
10.4.0.0/16	AS1	100	3	52	2025-02-11	172.21.0.6
10.3.0.0/16	AS5	100	4	50	2025-02-11	172.21.0.33
10.4.0.0/16	AS8	100	3	50	2025-02-11	172.21.0.45

AS7:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	3	20	2025-02-02	172.21.0.29
10.6.0.0/16	AS5	100	2	20	2025-02-03	172.21.0.37
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS4	100	3	20	2025-02-01	172.21.0.29
10.4.0.0/16	AS4	100	1	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS8	300	3	20	2025-02-01	172.21.0.42
10.1.0.0/16	AS4	200	3	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS5	400	5	80	2025-02-01	172.21.0.37

AS8:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.6.0.0/16	AS3	200	3	10	2025-02-01	172.21.0.25
10.7.0.0/16	AS3	200	3	10	2025-02-05	172.21.0.25
10.5.0.0/16	AS3	200	2	10	2025-02-03	172.21.0.25
10.4.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.8.0.0/16	Локально	500	0	0	2025-02-01	-
10.2.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.6.0.0/16	AS7	200	3	20	2025-02-01	172.21.0.41
10.3.0.0/16	AS3	200	1	10	2025-02-15	172.21.0.25
10.4.0.0/16	AS3	100	2	10	2025-02-03	172.21.0.25
10.6.0.0/16	AS6	50	1	20	2025-02-03	172.21.0.46

Определите сначала маршрут прохождения пакета из AS8 в AS6, а потом маршрут прохождения пакета в случае, если из-за аварии будет отключены связи между AS8 и AS3, а также между AS1 и AS6, а записи, соответствующие этим связям, будут исключены из таблиц.

В ответ запишите номера автономных систем через запятую, начиная с 8 и заканчивая 6, сначала для первого случая, а потом, через тире, для второго. Если маршрут построить нельзя (например, его нет или образуется цикл), вместо перечисления номеров автономных систем укажите NULL.

Пример ввода ответа: 8,10,12,6-8,10,1,2,6.

Ответ: 8,3,4,2,5,6-8,7,5,6

Решение:

В данной задаче необходимо аккуратно пройтись по таблицам маршрутизации и последовательно выбирать нужные строки.

Поскольку маршрут начинается в AS8, посмотрим таблицу маршрутизации этой автономной системы. В ней нужно выбрать те маршруты, в которых сеть назначения равна 10.6.0.0/16. Строки с такими маршрутами выделим зелёным цветом:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.6.0.0/16	**AS3**	**200**	**3**	**10**	**2025-02-01**	**172.21.0.25**

10.7.0.0/16	AS3	200	3	10	2025-02-05	172.21.0.25
10.5.0.0/16	AS3	200	2	10	2025-02-03	172.21.0.25
10.4.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.8.0.0/16	Локально	500	0	0	2025-02-01	-
10.2.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.6.0.0/16	AS7	200	3	20	2025-02-01	172.21.0.41
10.3.0.0/16	AS3	200	1	10	2025-02-15	172.21.0.25
10.4.0.0/16	AS3	100	2	10	2025-02-03	172.21.0.25
10.6.0.0/16	AS6	50	1	20	2025-02-03	172.21.0.46

При сравнении Local Preference из рассмотрения убирается маршрут в AS6, так как там значение меньше, а дальше при сравнении IGP Metric убирается маршрут в AS7, поскольку там значение больше. Остаётся маршрут в AS3.

Дальше аналогично рассмотрим маршруты из AS3:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.22
10.6.0.0/16	AS1	100	2	20	2025-02-01	172.21.0.10
10.6.0.0/16	**AS4**	**200**	**4**	**10**	**2025-02-03**	**172.21.0.22**
10.7.0.0/16	AS4	100	3	10	2025-02-04	172.21.0.22
10.3.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS8	100	3	10	2025-02-04	172.21.0.26
10.4.0.0/16	AS4	200	1	10	2025-02-04	172.21.0.22
10.4.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS1	100	1	20	2025-02-04	172.21.0.10

При сравнении Local Preference из рассмотрения убирается маршрут в AS1, и остаётся маршрут в AS4.

Рассмотрим маршруты из AS4:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.13
10.6.0.0/16	**AS2**	**100**	**3**	**5**	**2025-02-03**	**172.21.0.13**
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	AS7	100	3	10	2025-02-01	172.21.0.30
10.8.0.0/16	AS3	100	1	10	2025-02-01	172.21.0.21
10.6.0.0/16	AS3	100	3	5	2025-02-11	172.21.0.21
10.1.0.0/16	AS3	100	3	5	2025-02-01	172.21.0.21

При сравнении IGP Metric из рассмотрения убирается маршрут в AS7 (до этого все метрики равны), при сравнении даты появления маршрута из рассмотрения также убирается маршрут в AS3, так как он появился позже. Остаётся маршрут в AS2.

Рассмотрим маршруты из AS2:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.2.0.0/16	Локально	500	0	0	2025-02-01	-
10.3.0.0/16	AS1	100	2	20	2025-02-03	172.21.0.1
10.6.0.0/16	AS1	100	2	20	2025-02-10	172.21.0.1
10.7.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.14
10.8.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.6.0.0/16	**AS5**	**100**	**2**	**20**	**2025-02-03**	**172.21.0.18**
10.3.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.4.0.0/16	AS1	200	2	20	2025-02-03	172.21.0.1
10.4.0.0/16	AS4	100	1	20	2025-02-03	172.21.0.14
10.1.0.0/16	AS1	100	1	20	2025-02-03	172.21.0.1
10.1.0.0/16	AS4	200	3	20	2025-02-03	172.21.0.14

У двух нужных маршрутов все метрики до даты появления совпадают, а маршрут в AS5 старше маршрута в AS1, поэтому маршрут в AS1 убирается из рассмотрения, и остаётся маршрут в AS5.

Рассмотрим маршруты из AS5:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.4.0.0/16	AS2	100	2	10	2025-02-02	172.21.0.17
10.5.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	**AS6**	**100**	**1**	**20**	**2025-02-03**	**172.21.0.34**
10.8.0.0/16	AS7	100	3	20	2025-02-01	172.21.0.38
10.3.0.0/16	AS2	500	3	20	2025-02-01	172.21.0.17

10.2.0.0/16	AS2	100	3	20	2025-02-01	172.21.0.17
10.3.0.0/16	AS6	100	3	20	2025-02-01	172.21.0.34
10.4.0.0/16	AS7	100	2	20	2025-02-02	172.21.0.38
10.1.0.0/16	AS6	100	2	20	2025-02-02	172.21.0.34
10.1.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.17

В нём только один маршрут ведёт в нужную сеть, поэтому рассмотрим маршруты из AS6:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS1	100	3	20	2025-02-02	172.21.0.6
10.6.0.0/16	**Локально**	**500**	**0**	**0**	**2025-02-01**	**-**
10.4.0.0/16	AS5	100	3	51	2025-02-11	172.21.0.33
10.7.0.0/16	AS1	100	4	20	2025-02-03	172.21.0.6
10.8.0.0/16	AS1	100	3	20	2025-02-01	172.21.0.6
10.3.0.0/16	AS8	200	2	5	2025-03-08	172.21.0.45
10.4.0.0/16	AS1	100	3	52	2025-02-11	172.21.0.6
10.3.0.0/16	AS5	100	4	50	2025-02-11	172.21.0.33
10.4.0.0/16	AS8	100	3	50	2025-02-11	172.21.0.45

Здесь запрос в сеть назначения обрабатывается локально, т.е. текущей AS. Соответственно, пакет доставлен в AS6.

Итоговый маршрут получается следующим: **8,3,4,2,5,6**.

Дальше нужно так же пройтись по таблицам с учётом ситуации, когда из строя выходят каналы связи AS8 – AS3 и AS1 – AS6. Поскольку каналы связи двухсторонние, из таблиц маршрутизации удаляются записи не только из AS8 в AS3 и из AS1 в AS6, но и наоборот. Это сильно повлияет на маршрут пакета, поскольку исходно в первый раз пакет передаётся как раз из AS8 в AS3.

Рассмотрим маршруты из AS3 с учётом ограничений. Красным цветом выделены недоступные маршруты:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.6.0.0/16	AS3	200	3	10	2025-02-01	172.21.0.25
10.7.0.0/16	AS3	200	3	10	2025-02-05	172.21.0.25
10.5.0.0/16	AS3	200	2	10	2025-02-03	172.21.0.25
10.4.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.8.0.0/16	Локально	500	0	0	2025-02-01	-
10.2.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.6.0.0/16	**AS7**	**200**	**3**	**20**	**2025-02-01**	**172.21.0.41**
10.3.0.0/16	AS3	200	1	10	2025-02-15	172.21.0.25
10.4.0.0/16	AS3	100	2	10	2025-02-03	172.21.0.25
10.6.0.0/16	AS6	50	1	20	2025-02-03	172.21.0.46

Из двух оставшихся маршрутов, которые подходят, нужно выбрать маршрут в AS7, поскольку его значение Local Preference больше, чем у маршрута в AS6.

Рассмотрим маршруты из AS7:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	3	20	2025-02-02	172.21.0.29
10.6.0.0/16	**AS5**	**100**	**2**	**20**	**2025-02-03**	**172.21.0.37**
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS4	100	3	20	2025-02-01	172.21.0.29
10.4.0.0/16	AS4	100	1	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS8	300	3	20	2025-02-01	172.21.0.42
10.1.0.0/16	AS4	200	3	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS5	400	5	80	2025-02-01	172.21.0.37

Здесь подходит только один маршрут, в AS5. Рассмотрим маршруты из AS5:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.4.0.0/16	AS2	100	2	10	2025-02-02	172.21.0.17
10.5.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	**AS6**	**100**	**1**	**20**	**2025-02-03**	**172.21.0.34**
10.8.0.0/16	AS7	100	3	20	2025-02-01	172.21.0.38
10.3.0.0/16	AS2	500	3	20	2025-02-01	172.21.0.17
10.2.0.0/16	AS2	100	3	20	2025-02-01	172.21.0.17
10.3.0.0/16	AS6	100	3	20	2025-02-01	172.21.0.34
10.4.0.0/16	AS7	100	2	20	2025-02-02	172.21.0.38
10.1.0.0/16	AS6	100	2	20	2025-02-02	172.21.0.34

10.1.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.17

Таблица аналогична таблице в прошлом случае, поэтому рассмотрим маршруты из AS6:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS1	100	3	20	2025-02-02	172.21.0.6
10.6.0.0/16	**Локально**	**500**	**0**	**0**	**2025-02-01**	**-**
10.4.0.0/16	AS5	100	3	51	2025-02-11	172.21.0.33
10.7.0.0/16	AS1	100	4	20	2025-02-03	172.21.0.6
10.8.0.0/16	AS1	100	3	20	2025-02-01	172.21.0.6
10.3.0.0/16	AS8	200	2	5	2025-03-08	172.21.0.45
10.4.0.0/16	AS1	100	3	52	2025-02-11	172.21.0.6
10.3.0.0/16	AS5	100	4	50	2025-02-11	172.21.0.33
10.4.0.0/16	AS8	100	3	50	2025-02-11	172.21.0.45

Таблица, хоть и содержит удалённые маршруты, всё равно имеет только один маршрут для пакетов в нужную сеть, причём локальный, то есть пакет доставлен в AS6.

Итоговый маршрут получается следующим: **8,7,5,6**.

Ответ к задаче будет записываться следующим образом: **8,3,4,2,5,6-8,7,5,6**.

7. Технологии сортировки и фильтрации данных (2 балл) [Космический капитализм]

В свободное от работы время Николай Владимирович, ответственный за систему проведения олимпиад, разрабатывает новый космический симулятор, и на данный момент у него готова система инвентаря корабля с оборудованием, гиперпрыжки между двумя звёздными системами и торговля на планетах. Рассмотрим подробнее устройство симулятора.

Игровой мир состоит из звёздных систем, в каждой из которых есть несколько планет. Для перемещения корабля между звёздными системами необходим двигатель и топливный бак. На один гиперпрыжок корабль тратит количество топлива, равное расстоянию между звёздными системами в парсеках. Из этого следует, что корабль не может выполнить прыжок на расстояние больше, чем вместимость его топливного бака.

Владелец корабля имеет определённое количество галактических кредитов на счёте. На каждой планете представлены товары пяти видов: продовольствие, медикаменты, техника, роскошь и минералы. Для каждой планеты и каждого вида товара известно количество имеющегося товара в тоннах, стоимость покупки и продажи в галактических кредитах. Корпус корабля имеет заданную грузоподъёмность, и нельзя купить товаров больше, чем имеющаяся грузоподъёмность.

Сейчас Николай Владимирович пишет тест для проверки этой функциональности. Тестовый сценарий состоит из нескольких шагов:

1. Приобрести товар **одного вида** на текущей планете в некотором количестве;
2. Взлететь с планеты и выполнить гиперпрыжок **в другую систему**;
3. Приземлиться на выбранной планете и продать весь купленный на шаге 1 товар.

Для этого сценария есть отдельный файл сохранения, для которого известно количество товара на каждой планете, стоимость его покупки и продажи. Эти данные представлены в виде отдельной базы данных, которая доступна ниже.

Таблица stars хранит информацию о звёздных системах:

- id — уникальный идентификатор системы в рамках игрового мира;
- name — название системы.

Таблица planets хранит информацию о планетах:

- id — уникальный идентификатор планеты в рамках игрового мира;
- name — название планеты;
- star_id — идентификатор звёздной системы, к которой относится планета.

Таблица star_map хранит информацию о расстояниях между звёздными системами:

- star_from_id — идентификатор начальной звёздной системы;
- star_to_id — идентификатор звёздной системы назначения (обязательно отличается от star_from_id);
- distance — расстояние между этими звёздными системами в парсеках.

Таблица product_types хранит информацию о типах товаров:

- id — уникальный идентификатор типа товаров;
- name — название типа товаров.

Таблица prices хранит информацию о стоимости покупки и продажи каждого типа товаров на каждой из планет:

- planet_id — идентификатор планеты;
- product_type — идентификатор типа товаров;
- buy_price — стоимость покупки владельцем корабля одной тонны товара в галактических кредитах;
- sell_price — стоимость продажи владельцем корабля одной тонны товара в галактических кредитах;
- amount — количество товара в тоннах, доступное к покупке на данной планете.

В рамках данного сценария кредитов хватит для покупки любого количества любого товара на любой планете. Пока что в этом сценарии не определено, какой товар и в каком количестве нужно приобрести и на какой планете его продать. Николай Владимирович хочет подобрать эти параметры так, чтобы максимизировать полученную прибыль (разницу между стоимостью продажи и покупки). Известно, что в сохранении корабль находится на планете Раалито системы Хезе, грузоподъёмность корабля равна 80 тонн, а вместимости топливного бака хватит на прыжок расстоянием в 25 парсек. Как опытный программист, Николай Владимирович определил нужные параметры за несколько минут и дописал тест. Определите это вместе с ним и вы.

В качестве ответа введите одно целое число — прибыль в галактических кредитах, полученную после выполнения тестового сценария с найденными параметрами. Если окажется, что для тестового сценария нет ни одного набора параметров, который бы дал положительную прибыль, в качестве ответа введите 0.

Ответ: 6080

Решение:

Чтобы не писать длинный запрос с подзапросами, разобьём решение на три части.

Для начала нужно узнать идентификаторы планеты и соответствующей планетарной системы. Это можно выполнить одним запросом (заодно убедимся, что условие верное, и планета действительно находится в указанной системе):

```
SELECT p.id, p.name, s.id, s.name FROM planets p
INNER JOIN main.stars s ON s.id = p.star_id
WHERE p.name = 'Раалито';
```

Получится следующий вывод:

p.id	p.name	s.id	s.name
54	Раалито	10	Хезе

Теперь определим все планеты, которые находятся в системах, удалённых от текущей не более чем на 25 парсек:

```
SELECT p.* FROM star_map sm
INNER JOIN planets p ON star_to_id = p.star_id
WHERE star_from_id = 10 AND distance <= 25;
```

Получится следующая таблица:

id	name	star_id
1	Арнарик Гаудад	1
2	Рамгатру	1
3	Хартис	1
4	Проту Памперой	1
5	Калайна	1
6	Унашера	1
81	Гешши	17
82	Халгалла	17
83	Ципца	17
84	Эйя	17
85	Суджим	17
103	Оннд	22
104	Ушалла	22
105	Индерон	22
106	Энжа	22
107	Гаших	22
108	Уррагах	22
133	Зэмин	28
134	Рэдэа	28
135	Пунэ-анита	28
136	Хуку	28
167	Перегон	37
168	Глоткус	37
169	Мабэл	37
170	Дарзания	37
171	Ойкнур Бад	37
183	Альянони	40
184	Горик	40
185	Эгмер	40
186	Итниклей	40

299	Локон	66
300	Га-по	66
301	Иоханг	66
302	Шолоани	66
303	Пунисе	66

Третий запрос позволит построить таблицу с возможной прибылью от продажи того или иного товара на той или иной планете. Здесь необходимо вспомнить про ограничение грузоподъёмности корабля в 80 тонн:

```
SELECT pr.planet_id, pr.product_type, pr.sell_price, pr_from.buy_price,
pr_from.amount,
       MIN(pr_from.amount, 80) * (pr.sell_price - pr_from.buy_price) profit
FROM prices pr
INNER JOIN prices pr_from ON pr_from.planet_id = 54 AND pr_from.product_type =
pr.product_type
WHERE pr.planet_id IN (1, 2, 3, 4, 5, 6, 81, 82, 83, 84, 85, 103, 104, 105, 106, 107,
108, 133, 134, 135, 136, 167, 168, 169, 170, 171, 183, 184, 185, 186, 299, 300, 301,
302, 303)
ORDER BY profit DESC;
```

Первые 10 строк результата будут выглядеть так:

pr.planet_id	pr.product_type	pr.sell_price	pr_from.buy_price	pr_from.amount	profit
103	1	142	66	865	6080
183	1	136	66	865	5600
301	1	134	66	865	5440
186	1	133	66	865	5360
135	1	132	66	865	5280
133	1	129	66	865	5040
81	1	128	66	865	4960
83	1	128	66	865	4960
300	1	126	66	865	4800
82	1	125	66	865	4720

Получается, что максимальная прибыль равна 6080 галактических кредитов. Её можно получить, если купить 80 тонн минералов и продать их на планете Оннд.

Также можно было написать простые запросы вида **SELECT** * **FROM** [название таблицы] для всех указанных в условии таблиц и обработать данные вручную или воспользоваться библиотекой для доступа к SQL БД для используемого языка программирования, скачав представленный в условии файл с базой данных.

8. Технологии программирования (3 балла)

[Расписание станков]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	3 секунды
Ограничение по памяти	256 мегабайт

Известный завод ждет реорганизация — его собираются переоборудовать n новейшими станками. Перед тем, как эти станки будут установлены, требуется разработать интерфейс для обработки задач на этих станках.

После реорганизации наладчик завода будет распределять поступающие задачи между станками. У каждой задачи есть длительность. У каждого станка есть независимая очередь задач, причем новую задачу можно добавить либо строго в конец этой очереди, либо строго в начало, если это *срочная задача*.

Станки работают строго в порядке очереди, обрабатывая задачи подряд. Как только станок завершает задачу, он сразу же начинает работу над следующей в очереди, если такая есть. На переключение между задачами время не тратится.

Формально определим время начала работы над задачей как первый момент времени, после которого прогресс по задаче строго увеличится. Так, если в момент времени t на свободный станок приходят сначала обычная задача A и затем срочная задача U, временем начала U станет t, а временем начала A — момент завершения U.

Аналогично, время завершения работы над задачей — первый момент времени, когда прогресс по задаче достигает ее длительности. Так, если в момент времени t прогресс по задаче A достиг ее длительности, и станку поступил запрос на обработку срочной задачи U, временем завершения A все равно будет t.

Для большего понимания советуем после прочтения условия ознакомиться с иллюстрацией внизу.

Всего поддерживается четыре типа запросов.

1. Добавить задачу номер i длительностью d в конец очереди k-го станка. Если его очередь до этого была пустой, станок тут же начинает работу над добавленной задачей

2. Отменить задачу с номером i. Если эта задача еще не была начата, она удаляется из очереди, а порядок остальных задач в очереди не меняется.
   Если эта задача уже была начата, работа над ней тут же останавливается, и станок берет в работу следующую задачу из очереди. Если эта задача уже была завершена или такой задачи не было, запрос отмены игнорируется
3. Добавить *срочную задачу* с номером i длительностью d в начало очереди k-го станка. В этом случае работа над текущей задачей (если она есть) на k -м станке приостанавливается с сохранением прогресса, и в работу берется данная срочная задача.
   До момента завершения срочной задачи все запросы добавления или отмены новых задач к станку номер k **игнорируются** для экономии ресурсов. Иными словами, если в момент времени t поступил запрос добавления срочной задачи, все следующие запросы, поступающие к k-му станку в моменты времени с t **включительно** до $t + d$ **не включительно** будут проигнорированы.
   По завершении срочной задачи, если работа над какой-то обычной задачей в очереди была приостановлена, эта задача без задержек возвращается в обработку.
4. Вывести ожидаемый момент времени завершения работы k-го станка.
   Для станка без задач в очереди «ожидаемым» временем завершения его работы будем считать текущий момент времени.

Еще раз **обратите внимание**, что во время исполнения срочной задачи станок игнорирует только все запросы добавления и отмены задач (включая добавление другой срочной задачи). Про запрос отмены будем считать, что он идет к тому станку, в очереди которого лежит соответствующая задача. Запросы вывода ожидаемого момента завершения работы **не игнорируются**.

Вам необходимо помочь наладчику и вывести для каждой задачи время начала и завершения работы над ней.

Формат входных данных

В первой строке дано целое число T ($1 \le T \le 1000$) — количество наборов входных данных.

В первой строке описания набора входных данных через пробел даны два целых числа n и q ($1 \le n, q \le 10^5$) — количество станков и количество запросов. Гарантируется, что сумма n и сумма q по всем наборам входных данных обе не превосходят 10^5.

В следующих q строках описаны запросы к заводу в одном из следующих форматов:

1. «t *schedule* i (d) *on* k» — добавить задачу на k-й станок;
2. «t *cancel* i» — отменить задачу;
3. «t *urgent* i (d) *on* k» — добавить срочную задачу на k-м станке;
4. «t *expectation* k» — узнать текущее ожидаемое время завершения работы k-го станка.

Параметр t — момент совершения запроса (целое неотрицательное число от 0 до 10^9). Для всех запросов выполняется $1 \le k \le n$, $1 \le i \le 10^9$ и $1 \le d \le 10^9$.

Также гарантируется, что номера задач (i) уникальны и не повторяются, а запросы упорядочены по времени, то есть t каждого следующего запроса не меньше, чем t предыдущего.

Формат выходных данных

Выведите в отдельной строке текущее ожидаемое время работы соответствующего станка после каждого запроса типа «*expectation*».

Затем выведите по строке на каждую задачу, запрос на добавление которой не был проигнорирован. Задачи должны быть упорядочены по возрастанию их номеров. В каждой строке через пробел должны быть выведены три числа: номер задачи, время начала ее обработки и время конца ее обработки (или отмены, если задача была отменена до завершения).

Для задач, которые были отменены до начала работы над ними, считайте время начала равным времени первого запроса их отмены.

Пример

Стандартный ввод	**Стандартный вывод**
3 2 2 0 schedule 1 (10) on 1 4 schedule 2 (3) on 2 3 6 0 schedule 1 (5) on 1 0 schedule 2 (5) on 2 2 urgent 3 (5) on 1 6 expectation 1 6 expectation 2 6 expectation 3 2 17 0 schedule 1 (10) on 1 0 schedule 2 (6) on 2 2 schedule 3 (5) on 1 3 schedule 4 (100) on 1	1 0 10 2 4 7 10 6 6 1 0 10 2 0 5 3 2 7 113 16 1 0 10 2 0 12 3 5 5 4 13 14 5 12 13 6 5 11

Стандартный ввод	Стандартный вывод
3 schedule 5 (1) on 2 5 cancel 3 5 urgent 6 (6) on 2 8 cancel 6 9 cancel 6 10 cancel 6 10 schedule 7 (150) on 2 10 urgent 8 (3) on 1 11 schedule 9 (3) on 2 14 expectation 1 14 cancel 4 14 schedule 10 (2) on 1 14 expectation 1	8 10 13 9 13 16 10 14 16

Замечание

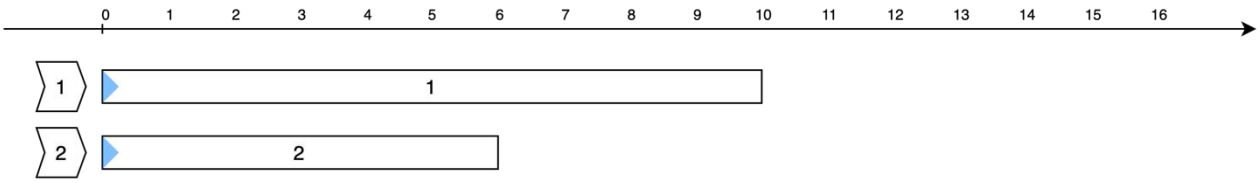
В первом примере из условия обе задачи будут обработаны по расписанию: с 0 до 10 и с 4 до 7 соответственно.

Во втором примере:

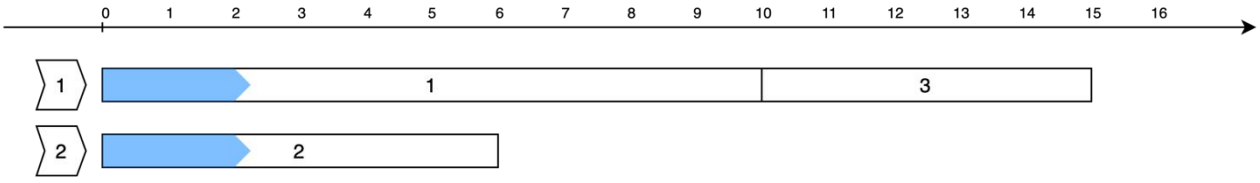
1. первая задача должна обрабатываться на первом станке в период $[0, 5]$;
2. вторая задача должна обрабатываться на втором станке в период $[0, 5]$;
3. в момент времени 2 срочная задача номер 3 добавляется на первый станок; она будет обрабатываться в период $[2, 7]$;
4. в момент времени 6 первому станку нужно еще 4 единицы времени на завершение задач 4 и 1; второй станок завершил работу в момент времени 5, а третий станок вообще не начинал работу — для них время ожидания до завершения работы равно 0.

Подробная иллюстрация к третьему примеру приведена ниже. Здесь синим обозначен прогресс по задачам, зеленым — завершенные задачи, красным — отмененные, а градиентом — срочные. Все интервалы короче 1 единицы времени (см. отмененную задачу 3) стоит воспринимать как интервалы длительностью 0.

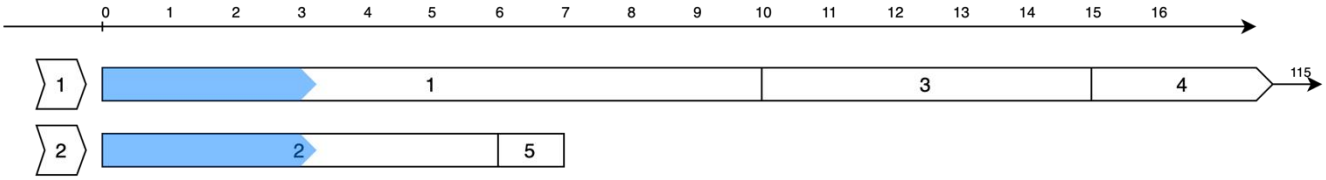
0 schedule 1 (10) on 1
0 schedule 2 (6) on 2



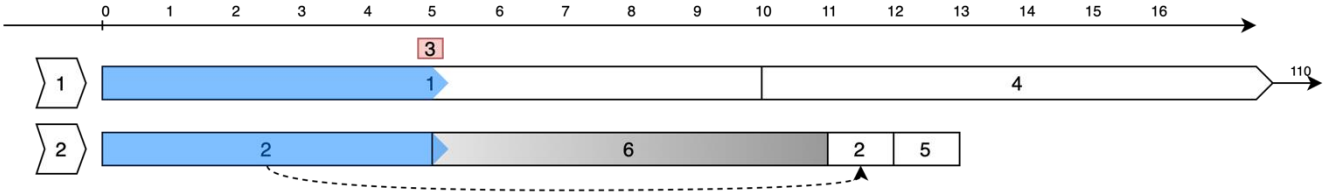
2 schedule 3 (5) on 1



3 schedule 4 (100) on 1
3 schedule 5 (1) on 2

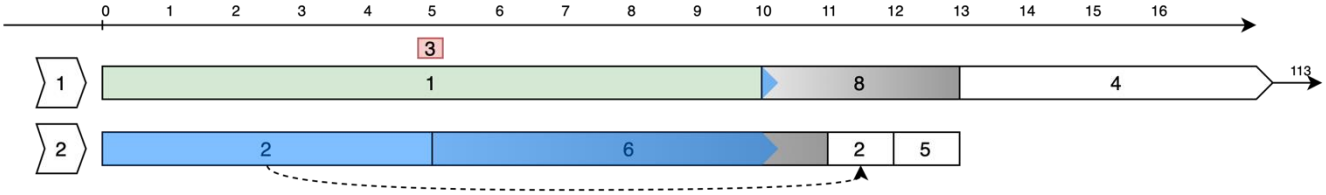


5 cancel 3
5 urgent 6 (6) on 2

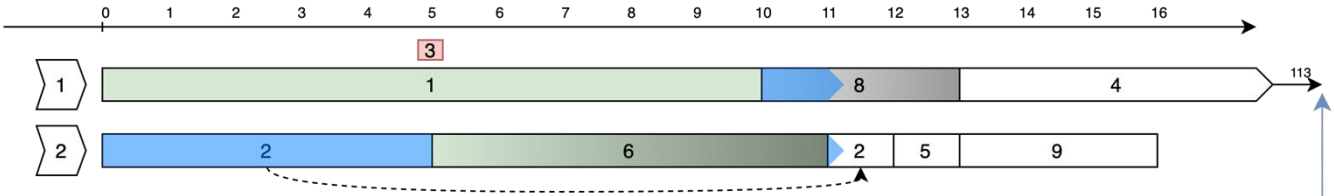


8 cancel 6
9 cancel 6
10 cancel 6
10 schedule 7 (150) on 2
10 urgent 8 (3) on 1

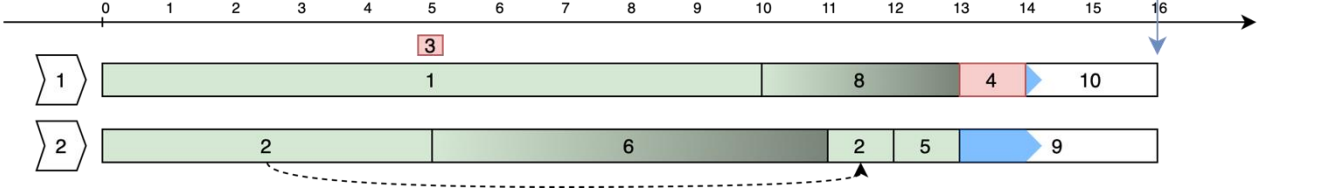
игнорируются, так как второй станок занят срочной задачей



11 schedule 9 (3) on 2



14 expectation 1
113
14 cancel 4
14 schedule 10 (2) on 1
14 expectation 1
16



Решение:

Это по большей части задача на реализацию, требующая при этом воспользоваться идеей ленивых вычислений. Разберем ее по частям.

Начнем с того, что, собственно, требуется хранить:

1. Для каждой задачи необходимо знать время начала и завершения ее обработки. Поскольку эти значения могут быть выставлены как последовательно при обработке, так и в любой момент времени при отмене, имеет смысл просто иметь доступ к задаче по ее номеру за $O(1)$: заведем словарь (unordered map, hash map) $jobs$, сопоставляющий номера задач и сами задачи.
2. Также для задачи полезно понимать, срочная ли она ($job.is\_urgent$), и какой прогресс по ней уже выполнен ($job.progress$). Ну и саму длительность задачи $job.duration$ тоже запомним отдельным полем. Определим $job.remaining()$ как 0, если она отменена, и как разницу между длительностью и прогрессом иначе.
3. Для каждого станка в любой момент времени может понадобиться узнать время его остановки. Мы будем хранить это отдельным полем $machine.finish$ для каждого станка, так как это самый эффективный способ быстро пересчитывать это значение при добавлении или отмене задач.
4. Разумеется, для каждого станка надо хранить указанную в условии очередь задач $machine.queue$.
5. И для обновления $machine.finish$ надо хранить словарь, сопоставляющий задачи станкам, на которых они обрабатываются.

Однако этого недостаточно. В этой задаче необходимы ленивые вычисления по одной причине: отмененная задача может находиться в середине очереди, из-за чего ее явное удаление оттуда может быть достаточно долгой операцией. Можно было бы в качестве альтернативы хранить упорядоченное множество вместо очереди, однако это замедлит и усложнит решение.

Идея ленивых вычислений в следующем: вместо реального удаления задач будем просто помечать задачу удаленной ($job.is\_cancelled$), а удалим ее из очереди, когда очередь до нее дойдет. Вторая идея — обрабатывать задачи не в режиме реального времени, а по необходимости. Действительно, станки независимы, поэтому можно проделать всю работу станка между предыдущим и текущим моментом времени только тогда, когда есть необходимость, например, понять где сейчас начало очереди. Поэтому будем дополнительно для каждого станка хранить последний момент времени, на котором мы «остановились» $machine.t$.

Тогда заведем следующие методы:

```
func (machine) work_until(time):
    while machine.t < time and len(machine.queue) > 0:
        job = machine.queue[0]
        if job.is_cancelled:
            machine.queue.popleft()
            continue

        machine.process_job(job, time)
        if job.remaining() == 0:
            machine.queue.popleft()

    machine.t = time
```

Иными словами, при необходимости в момент времени t узнать актуальное состояние станка, выполним всю его работу до этого момента: будем увеличивать прогресс по работам по очереди, пока не дойдем до текущего момента. При этом $process\_job$ выглядит следующим образом:

```
func (machine) process_job(job, time):
    rem = min (time - machine.t, job.remaining())
    if job.start is None:
        job.start = machine.t

    job.progress += rem
    machine.t += rem
    if job.remaining():
        job.finish = machine.t
```

Здесь мы увеличиваем прогресс по задаче на минимум из того, сколько осталось времени до ее завершения и до достижения момента $time$. Не забываем при этом выставить начало и конец обработки задачи в соответствии с определениями из условия.

Остается только аккуратно обработать все запросы. Перед выполнением запроса к станку вызываем для него $work\_until$, проверяем, что запрос не игнорируется, добавляем задачу в очередь и увеличиваем $machine.finish$ на ее длительность. При отмене задачи просто помечаем задачу отмененной, выставляем ей конец обработки и, возможно, начало, после чего уменьшаем $machine.finish$ для соответствующего станка на $job.remaining()$.

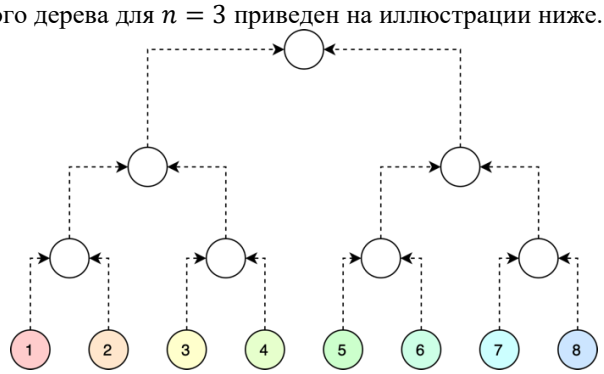
В итоге получаем решение, работающее за время $O(n + q)$, хотя в некоторых тестах и решение за $O(nq)$ могло пройти по времени.

9. Технологии программирования (4 балла)

[Бинарная Сила]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	4 секунды
Ограничение по памяти	256 мегабайт

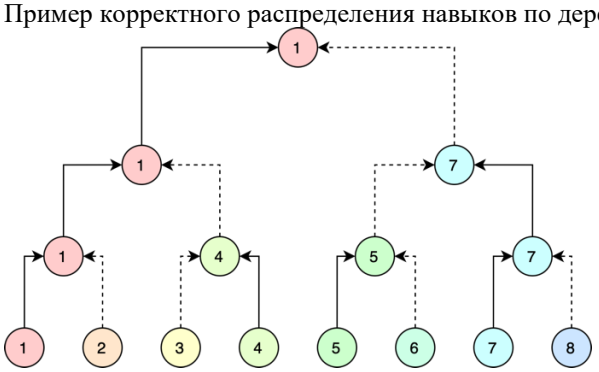
Вы играете в игру «Бинарная Сила» и управляете персонажем, у которого есть $d = 2^n$ навыков, пронумерованных 1 до d. Эти навыки расположены на листьях полного двоичного дерева высоты n, изначально все навыки имеют уровень 1. Пример такого дерева для $n = 3$ приведен на иллюстрации ниже.



После этого вы начинаете прокачивать навыки следующим образом.

- Навыки прокачиваются посредством заполнения двоичного дерева снизу вверх.
- Для очередной вершины дерева вы должны выбрать и записать в нее один из двух навыков, записанных в непосредственных детях этой вершины (на рисунке из детей в родителя ведут стрелки).
- *Уровнем* навыка считается число вершин, в которых выбран этот навык.

Пример корректного распределения навыков по дереву для $n = 3$ приведен ниже.



В этом примере первый навык имеет уровень 4, седьмой – уровень 3, четвертый и пятый – уровень 2, а второй, третий, шестой и восьмой не были прокачаны ни разу, поэтому остались на уровне 1.

Кроме прокачки персонажа, в игре есть m различных квестов, с помощью которых можно получать монетки. Квесты активируются после того, как все дерево навыков было заполнено.

Квесты бывают трех типов:

1. «$less\ x_i\ k_i\ s_i$» – вы получите s_i монет, если уровень навыка с номером x_i окажется **строго меньше** k_i.
2. «$exact\ x_i\ k_i\ s_i$» – вы получите s_i монет, если уровень навыка с номером x_i окажется **равен** k_i.
3. «$less\ x_i\ k_i\ s_i$» – вы получите s_i монет, если уровень навыка с номером x_i окажется **строго больше** k_i.

Так как монеты – очень ценный ресурс в игре «Бинарная Сила», вы хотите узнать максимальное количество монет, которое возможно получить с помощью имеющихся квестов после улучшения всех навыков.

Формат входных данных

Каждый тест состоит из нескольких независимых наборов входных данных. Первая строка содержит одно целое число t – количество наборов входных данных ($1 \le t \le 10^4$). Далее следует описание наборов входных данных.

Каждый набор начинается со строки, содержащей два целых числа n и m – высоту дерева навыков и количество квестов соответственно ($1 \le n \le 15$; $0 \le m \le 50\,000$). Число навыков при этом равно $d = 2^n$.

Далее следуют m строк, i-я из которых содержит четыре целых числа t_i, x_i, k_i, s_i – тип квеста и его описание ($1 \le t_i \le 3$; $1 \le x_i \le d$; $1 \le k_i \le n$; $1 \le s_i \le 10^9$). Типы квестов следуют в том же порядке, в котором они перечислены в условии: $t_i = 1$ соответствует квесту типа «$less$», $t_i = 2$ – квесту типа «$exact$» и $t_i = 3$ – квесту типа «$more$».

Гарантируется, что сумма d по всем наборам входных данных не превосходит 2^{16} и сумма m по всем наборам входных данных не превосходит $50\,000$

Формат выходных данных

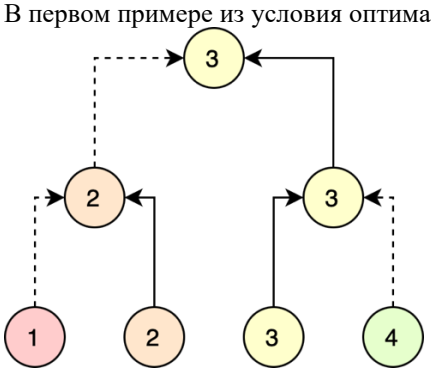
Для каждого набора выходных данных в отдельной строке выведите единственное число – максимальное количество монет, которые можно заработать.

Пример

Стандартный ввод	Стандартный вывод
4 2 5 2 1 1 10 2 2 2 100 1 2 2 5 3 3 1 15 2 3 1 10 1 3 2 2 1 5 2 1 1 10 3 1 1 3 10 6 2 1 1 1 2 2 2 1 2 4 3 1 2 8 4 1 2 16 5 1 2 32 6 1 10 4 1 6 3 5 1 6 4 7 2 6 5 11 3 6 1 3	125 10 6 15

Замечание

В первом примере из условия оптимальное распределение навыков выглядит следующим образом:



Действительно, мы получаем 100 монет за то, что второй навык имеет уровень **ровно** два, заодно получаем еще 10 монет за то, что первый имеет уровень **ровно** один. В этом случае третий квест не может быть выполнен, а из двух оставшихся мы получаем наибольшую награду (15 монет), если третий навык имеет уровень больше 1.

В третьем примере из условия достаточно для каждой вершины выбрать навык из правого ребенка. Тогда уровень навыка с номером 2^i будет в точности равен $i + 1$, и все квесты будут выполнены.

Решение:

В этой задаче можно было попробовать использовать жадные идеи — стараться сначала выполнять более дорогие квесты, но они не дали бы оптимальный ответ в большой части случаев. Так что будем оптимизировать полный перебор всех вариантов с помощью динамического программирования.

Динамическое программирование позволяет нам заметить, что если бы мы делали полный перебор, то при выборе навыка в очередной вершине нам были бы важны только значения навыков в ее детях, но не ниже. Соответственно, достаточно не перебирать все возможные распределения навыков по дереву, а только для каждой пары из вершины и ее возможного навыка определить оптимальный ответ на ее поддереве. Введем $dp[v][i]$ — оптимальный ответ на поддереве v, если в ней стоит навык i.

Теперь надо научиться пересчитывать эти значения. Порядок пересчета понятен — просто считаем снизу вверх по дереву, тогда при вычислении $dp[v]$ значения dp для ее детей уже посчитаны. Теперь заметим, что случаи, когда i находится в левом поддереве v, и в правом, симметричны, поэтому далее мы разберем только первый случай, а второй будет аналогичен.

Для удобства реализации пронумеруем все вершины сверху вниз и слева направо, начиная с нуля. Тогда несложно заметить, что дети вершины v будут иметь номера $2v + 1$ и $2v + 2$. При этом номер слоя вершины v будет равен $\lfloor log_2(v + 1) \rfloor$, что мы обозначим за $layer(v)$, а номера листьев в ее поддереве (в нумерации с нуля) будут лежать в интервале от $\min(v) = (v + 1) * 2^{n-layer(v)} - 2^n$ до $\max(v) = \min(v) + 2^{n-layer(v)}$ не включительно. Скажем, что $dp[v][i]$ отвечает за лист номер $\min(v) + i$.

При этом также сразу оценим, что сумма количества листьев по всем поддеревьям дерева имеет размер порядка $2^n * n$, так как каждый лист входит в n поддеревьев. Поэтому если мы будем пересчитывать $dp[v][i]$ за $O(1)$, полная асимптотика времени решения будет равна $O(2^n * n)$.

Пусть $i \in [min(v), \frac{min(v)+max(v)}{2})$, то есть находится среди левой половины листьев в поддереве v. Тогда $dp[v][i]$ подразумевает, что в вершине $2v + 1$ (в левом ребенке) также был выбран навык номер i. При этом в правом ребенке может быть выбран любой навык, поэтому

$$dp[v][i] = dp[2v + 1][i] + \max_{j \in [\min(2v+2), \max(2v+2))} dp[2v + 2][j] + REWARD$$

Здесь $REWARD$ означает количество монет, которое мы получим за размещение i-го навыка в v-й вершине. Давайте считать, что все квесты типа «$less$» изначально выполнены (то есть в базе динамики положим значения, равные сумме монет за все квесты типа «$less$» с уровнем больше 0), тогда за такое действие мы получим:

- минус число монет за квесты «$less$» для i-го навыка и уровня, соответствующего v (действительно, до этого ограничение квеста выполнялось, а теперь перестало);
- монеты за квесты «$exact$» для i-го навыка и уровня, соответствующего v;
- монеты за квесты «$more$» с тем же уровнем навыка по аналогичным причинам.

Чтобы быстро считать это число монет, заранее сгруппируем все квесты по навыкам, а в каждой группе — по уровню. И для каждого уровня сложим монеты из квестов «$exact$» и «$more$» и вычтем монеты за квесты «$less$». Это и будет значение $REWARD$, используемое выше.

Остается только заметить, что $\max_j dp[v][j]$ тоже можно поддерживать во время пересчета dp в отдельном массиве $\max\_dp$, добавив $O(1)$ времени на каждый пересчет. Тогда в сумме весь пересчет отработает за время $O(2^n * n)$, а ответ в конце будет находиться в $\max\_dp[0]$.

Отборочный этап. Первый тур (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (3 балла) [Слишком длинный период]

Петя, будучи уже опытным пользователем суперкомпьютера, решил посчитать на нём различные степени некоторого целого положительного числа a в порядке возрастания, начиная с первой. Все результаты он выводил в системе счисления с основанием $C = 61$, причём выводил только последние три цифры получаемых чисел, при необходимости дописывая ведущие нули. В какой-то момент Петя заметил, что вывод суперкомпьютера стал повторяться, то есть последовательность оказалась периодической. Ему стало интересно, какую максимальную длину периода он сможет получить, он дописал программу и стал ждать результат.

Пока Петя ждал, к нему в гости зашёл Вася. Петя рассказал ему про задачу, и через несколько минут Вася сказал ему ответ. Петя с удивлением посмотрел на получившееся число, однако спустя полчаса вычислений на суперкомпьютере Петя получил свой ответ, который сошёлся с ответом Васи. Попробуйте и Вы определить, какова максимальная длина периода такой последовательности.

Ответ: 223260

2. Кодирование информации. Системы счисления (2 балла) [Бесконечная дробь]

Найдите максимальное рациональное число R, меньшее 1 такое, что если его сложить с числом $(1/255)_{10}$ и перевести результат в четверичную систему счисления, то в дробной части будут встречаться только цифры 1. В ответе укажите несократимую дробь в виде m/n, представив числитель и знаменатель в десятичной системе счисления, например, 9/41.

Найдите такое минимальное основание позиционной системы счисления C, при котором выражение $0{,}2(12)_C \cdot 110_{10}$ будет натуральным числом.

В ответе укажите натуральное число. Если такого основания не существует, в ответе напишите NULL.

Ответ: 4

3. Кодирование информации. Количество информации (2 балла) [Цветные номера]

В мешке лежит набор из $A \cdot 13$ шаров. Каждый шар окрашен в один из A различных цветов, и шары каждого цвета пронумерованы от 1 до 13 (то есть все шары различны).

Дима наудачу достал $2A$ шаров из мешка. Известно, что сообщение «в выбранном наборе встречаются шары только с двумя различными номерами» несёт в себе минимум 23 бит информации. Определите наименьшее возможное количество цветов A. В ответе укажите целое число.

Ответ: 4

4. Кодирование информации. Количество информации. Объём информации (3 балла) [Формат Мити]

В одной из аудиторий университета стоит фортепиано, доступное для всех желающих. Митя, будучи выпускником музыкальной школы, каждую неделю играет на нём композиции из своего сборника. Образцы композиций хранятся у него на компьютере в специальном формате.

Аудиофайл формата Мити состоит из команд, которые так или иначе влияют на звучание композиции, в отличие от большинства других форматов, хранящих оцифрованный аналоговый звук. Для записи файла Митя использует две команды – "зажать клавишу" и "отпустить клавишу". Каждая из двух команд занимает 32 бита, разделенных на следующие фрагменты:

- 8 бит на хранение времени, которое нужно отсчитать с момента прошлой команды, прежде чем выполнится текущая;
- 4 бита на тип команды;
- 4 бита на номер инструмента, которому предназначается команда;
- 8 бит на номер ноты (от 0 до 127);
- 8 бит на силу нажатия (от 0 до 255; не несет информации, если выполняется команда "отпустить клавишу").

Такой формат позволяет проигрывать две и более ноты одновременно.

К Мите часто приходит Женя, увлекающийся сжатием данных. Он знает, как хранятся композиции Мити, и хочет предложить ему альтернативный вариант кодирования команд, в котором для кодирования каждого фрагмента команды будет использоваться минимально возможное одинаковое для каждого такого фрагмента целое количество бит, при этом не несущие для данной команды информации фрагменты исключаются из команды. В том числе Женя обратил внимание, что Митя умеет играть только на одном музыкальном инструменте, и подумал, где еще он может уменьшить количество бит для хранения параметра, но точно известно, что он не стал изменять формат фрагмента для хранения времени между командами.

Также к Мите иногда приходит Настя, которая записывает игру Мити на диктофон. Она всегда включает запись в тот момент, когда Митя начинает играть композицию, и выключает запись ровно в тот момент, когда он заканчивает её играть. Диктофон записывает звук в формате моно с частотой дискретизации 22 кГц и 65536 уровнями квантования.

На этой неделе ребята решили провести эксперимент. Митя подобрал композицию, в которой в каждый момент времени играется ровно одна нота и все ноты играются без пауз. Женя заранее конвертировал запись Мити в свой вариант формата, а Настя записала эту композицию на диктофон, после чего ребята сравнили размеры получившихся файлов.

Известно, что в композиции есть только ноты длительностью 1/8 секунды, 1/4 секунды и 1/2 секунды. Все ноты игрались непрерывно, и в файле Мити для этой композиции команда "зажать клавишу" для новой ноты выполняется сразу же после команды "отпустить клавишу" для текущей ноты.

Определите длительность композиции, сыгранной Митей, если известно, что запись Жени меньше записи Мити на 2223 байт, а запись Насти больше записи Мити на 4306072 байт. Ответ запишите в секундах, при необходимости округлите ответ вниз.

Ответ: 98

5. Основы логики. Анализ логических функций (2 балла) [Стрелочки]

Даны две логические функции:

$$X(a,b,c,d,e,f,g,h) = \Big(\overline{\Big(\big(d \to \big(b \to \big(c \to (\overline{e} \to \overline{a})\big)\big)\big) \to \overline{f}\Big)} \to \overline{\Big(\big(c \to \big(d \to \big(a \to (b \to e)\big)\big)\big) \to \overline{g}\Big)}\Big) \to \overline{\Big(\big(\overline{e} \to \big(b \to \big(a \to (d \to \overline{c})\big)\big)\big) \to \overline{h}\Big)}$$

$$Y(a,b,c,d,e,f,g,h) = \overline{\Big(\big(d \to \big(b \to \big(c \to (\overline{e} \to \overline{a})\big)\big)\big) \to \overline{f}\Big)} \to \Big(\overline{\Big(\big(c \to \big(d \to \big(a \to (b \to e)\big)\big)\big) \to \overline{g}\Big)} \to \overline{\Big(\big(\overline{e} \to \big(b \to \big(a \to (d \to \overline{c})\big)\big)\big) \to \overline{h}\Big)}\Big)$$

Обратите внимание, что указанные логические функции могут иметь длинную запись, приводящую к появлению горизонтальной полосы прокрутки.

Упростите выражение $\overline{X} \to Y$, если известно, что выражение $a \to \big(b \to \big(c \to (d \to e)\big)\big)$ является истинным.

*Комментарий по вводу ответа: операнды вводятся строчными **латинскими** буквами; логические операции обозначаются, соответственно, как **not**, **and** и **or**. Результат упрощения может содержать только переменные a, b, c, d, e, f, g и h. Скобки используются только для изменения порядка выполнения операций. Если порядок выполнения операций очевиден из их приоритетов – дополнительное использование скобок считается ошибкой.*
При однозначном ответе – истинный ответ обозначается как 1, а ложный как 0.

Пример записи ответа: (a or not b) and c

Ответ: not f or not g or h || not f or h or not g || not g or not f or h || not g or h or not f || h or not f or not g || h or not g or not f

6. Основы логики. Упрощение логического выражения (1 балла) [Сплошные отрицания]

Упростите логическое выражение или укажите его результат (при его однозначности). Результат упрощения может содержать только операции инверсии, конъюнкции и дизъюнкции.

$$\Big(\big((A|B)|(\overline{A} \vee \overline{B})\big)\,|\,\big((\overline{(C \wedge D)})|(C|D)\big)\Big)\,|\,\big(((C \downarrow C) \downarrow (D \downarrow D))|((\overline{A} \vee \overline{B})|(\overline{(A \wedge B)}))\big)$$

Примечание:

Логическая операция $X|Y$ (штрих Шеффера) имеет следующую таблицу истинности:

X	Y	$X\|Y$
0	0	1
0	1	1
1	0	1

1	1	0

Логическая операция $X \downarrow Y$ (стрелка Пирса) имеет следующую таблицу истинности:

*Комментарий по вводу ответа: операнды вводятся большими **латинскими** буквами; логические операции обозначаются, соответственно как **not**, **and** и **or**.*

Скобки используются только для изменения порядка выполнения операций. Если порядок выполнения операций очевиден из их приоритетов – дополнительное использование скобок считается ошибкой.

При однозначном ответе – истинный ответ обозначается как 1, а ложный как 0.

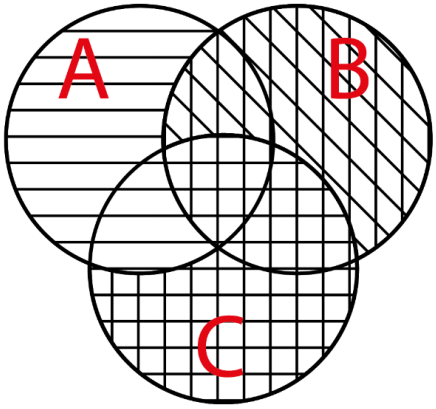
Пример записи ответа: (A or not B) and C

Ответ: A and B and C and D || A and B and D and C || A and C and B and D || A and C and D and B || A and D and B and C || A and D and C and B || B and A and C and D || B and A and D and C || B and C and A and D || B and C and D and A || B and D and A and C || B and D and C and A || C and A and B and D || C and A and D and B || C and B and A and D || C and B and D and A || C and D and A and B || C and D and B and A || D and A and B and C || D and A and C and B || D and B and A and C || D and B and C and A || D and C and A and B || D and C and B and A

7. Основы логики. Синтез логического выражения (1 балл)

[Три штриха]

Дана диаграмма Эйлера-Венна для трех логических переменных: A, B и C.



Известно, что логическая функция $X(A,B,C)$ истинна только в области, имеющей горизонтальную штриховку, логическая функция $Y(A,B,C)$ истинна только в области, имеющей вертикальную штриховку, а логическая функция $Z(A,B,C)$ истинна только в области, имеющей диагональную штриховку. Найдите логическую функцию $Z(X,Y)$. Если таких функций несколько, запишите любую из них. В ответе запишите найденную функцию в максимально упрощенном виде. Запись может содержать только логические переменные X и Y и операции конъюнкции, дизъюнкции и инверсии и не может содержать скобки. Если таких функций не существует, запишите в ответ NULL.

Ответ: not X and Y || Y and not X

8. Алгоритмизация и программирование. Формальный исполнитель (2 балла)

[Рекурсивное закрашивание]

Разбирая старые задачи олимпиады, Петя наткнулся на алгоритм рекурсивного закрашивания растрового изображения. У Пети есть черно-белое (bitmap) изображение размером 13 на 13 пикселей. На изображении присутствует замкнутый контур, как приведено на рисунке. Пиксели внутри контура пронумерованы.

Традиционно для компьютерной графики, система координат имеет начало в верхнем левому углу, ось X направлена слева направо, а ось Y – сверху вниз.

Алгоритм рекурсивного закрашивания заключается в рекурсивном вызове процедуры «Закрасить», которой передаются два параметра – координаты X и Y пикселя.

Процедура *Закрасить(X, Y)*, может быть описана следующим образом:

3. Если цвет пикселя с координатами (X, Y) белый, то:
   a. Изменить цвет пикселя с этими координатами на черный;
   b. Вызвать процедуру *Закрасить(X+1, Y)*;

c. Вызвать процедуру *Закрасить(X, Y+1)*;
d. Вызвать процедуру *Закрасить(X-1, Y)*;
e. Вызвать процедуру *Закрасить(X, Y-1)*;

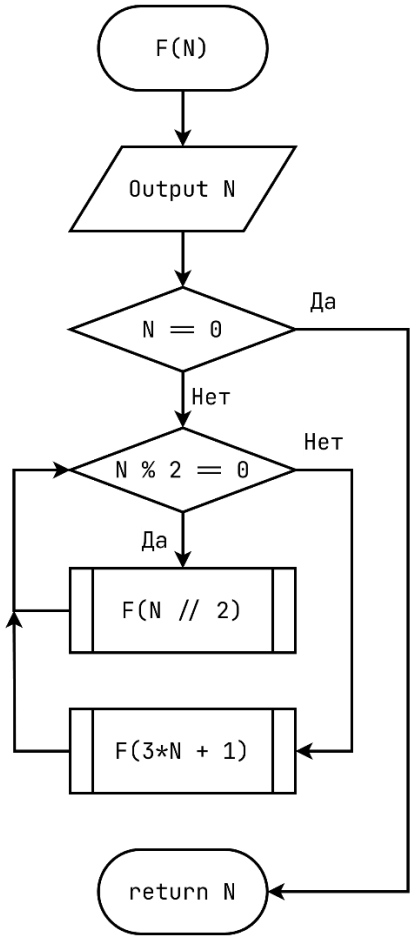
4. Иначе завершить процедуру.

Известно, что последний закрашенный пиксель, перед завершением процедуры, имел номер 29. Сколько существует пикселей внутри контура, в которых можно исходно вызвать процедуру «Закрасить» так, чтобы получить такой результат? В ответе укажите целое число.

Ответ: 47

9. Алгоритмизация и программирование. Блок-схема, прямая задача (3 балла) [Проверка гипотезы]

Дана блок-схема алгоритма, реализованного в виде рекурсивно вызываемой функции:

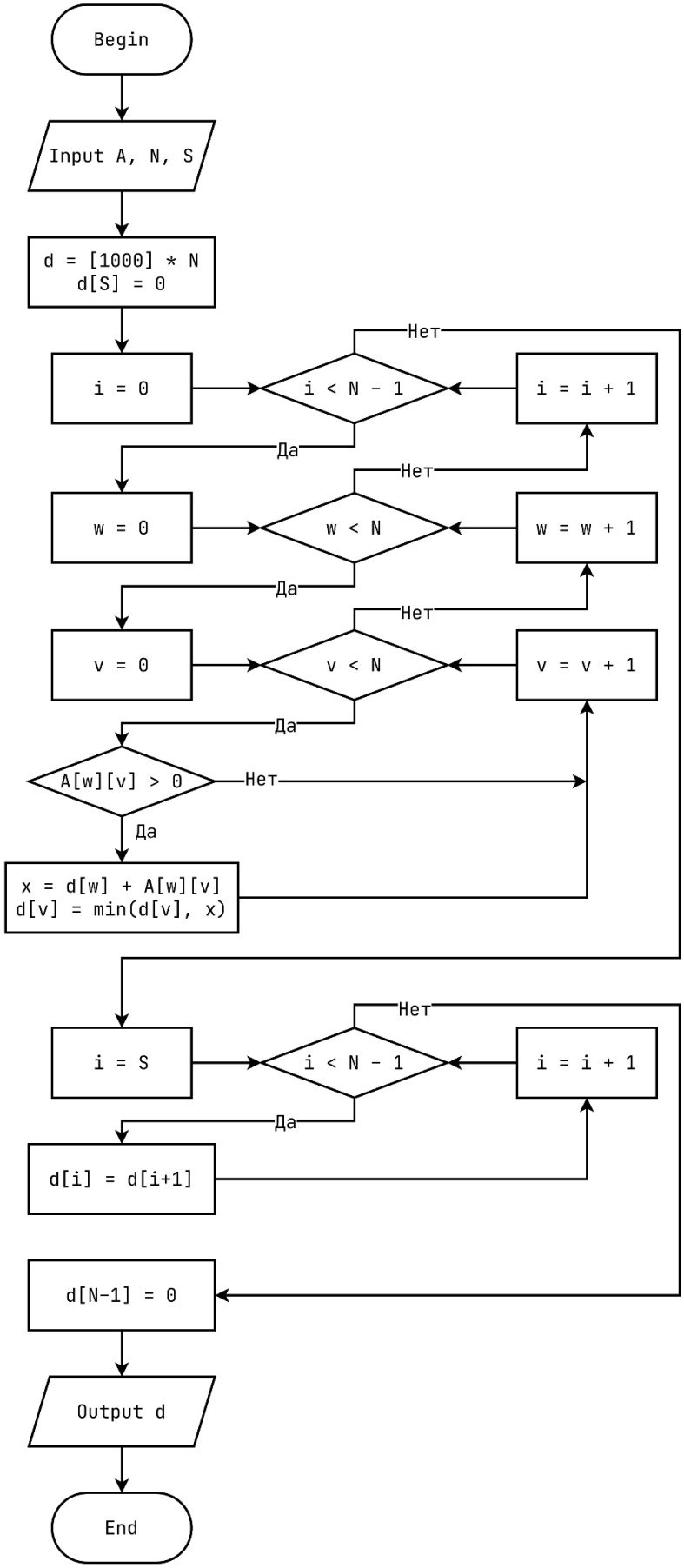


Определите сумму первых 10^{10} чисел, которые выведутся при запуске алгоритма с $N = -3159$.

Ответ: -900557262045

10. Алгоритмизация и программирование. Блок-схема, обратная задача (1 балл) [Агрегация матрицы]

Дана блок-схема алгоритма, обрабатывающего квадратную матрицу A размера $N \times N$ и выводящего одномерный массив d с N элементами:



Ричард изучает этот алгоритм, передавая ему разные матрицы на вход. Одна из таких матриц указана ниже:

0	4	0	4	0	0	0	2	0
0	0	0	0	0	0	8	10	0
0	13	0	1	0	0	0	0	5

0	7	0	0	0	6	0	0	0
4	0	0	0	0	0	0	0	0
11	0	18	0	5	0	0	0	3
0	0	4	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Определите значение переменной S, которое нужно передать на вход вместе с матрицей, чтобы вывелся следующий массив: [4, 16, 4, 15, 10, 12, 2, 13, 0]. Если таких значений несколько, введите любое из них. Если такого значения нет, введите NULL.

Примечание. Функция `min` вычисляет минимум из двух чисел. Запись вида `[1000] * N` означает, что создаётся массив, состоящий из N повторений числа 1000. Нумерация элементов массива начинается с 0. При обращении к элементам матрицы первый индекс означает номер строки, а второй – номер столбца с нумерацией от 0.

Ответ: 0

Отборочный этап. Второй тур (приведен один из вариантов заданий)

1. Электронные таблицы. Адресация ячеек и вычисления (1 балл)

[Усредняем и фиксируем]

Ниже приведен фрагмент электронной таблицы в режиме отображения формул. Известно, что в ячейке A1 так же записана формула, в которой задействованы ячейки B1:F1. Эту формулу скопировали во все ячейки A2:A20.

	A	B	C	D	E	F	G
1		22	32	89	14	58	=СРЗНАЧ(A1:F1)
2		18	77	86	51	48	=СРЗНАЧ(A2:F2)
3		25	23	9	14	44	=СРЗНАЧ(A3:F3)
4		24	94	69	89	14	=СРЗНАЧ(A4:F4)
5		15	7	5	40	98	=СРЗНАЧ(A5:F5)
6		41	83	45	14	97	=СРЗНАЧ(A6:F6)
7		89	60	27	71	33	=СРЗНАЧ(A7:F7)
8		100	3	97	37	73	=СРЗНАЧ(A8:F8)
9		66	3	7	60	19	=СРЗНАЧ(A9:F9)
10		14	43	71	49	3	=СРЗНАЧ(A10:F10)
11		43	22	76	21	3	=СРЗНАЧ(A11:F11)
12		75	12	78	32	73	=СРЗНАЧ(A12:F12)
13		71	14	94	50	36	=СРЗНАЧ(A13:F13)
14		99	27	59	16	59	=СРЗНАЧ(A14:F14)
15		72	49	62	47	100	=СРЗНАЧ(A15:F15)
16		69	71	93	25	47	=СРЗНАЧ(A16:F16)
17		42	35	5	84	49	=СРЗНАЧ(A17:F17)
18		91	25	41	100	13	=СРЗНАЧ(A18:F18)
19		28	70	47	22	43	=СРЗНАЧ(A19:F19)
20		94	2	78	54	72	=СРЗНАЧ(A20:F20)
21	=СУММ(A1:A20)						=СУММ(G1:G20)

Также известно, что значение ячейки A1 совпадает со значением ячейки G1, и значение ячейки A21 совпадает со значением ячейки G21.

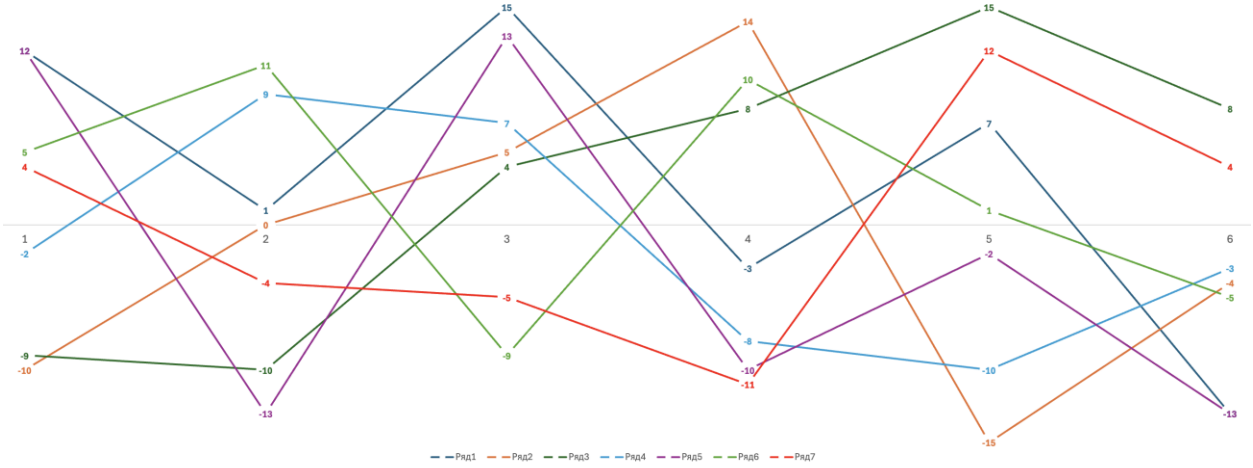
Найдите значение ячейки A21.

Ответ: 972

2. Электронные таблицы. Графики и диаграммы (2 балла)

[Почти сорвавшийся эксперимент]

Вася провёл эксперимент, в конце которого у него получилось 6 массивов данных: a_1, a_2, a_3, a_4, a_5, a_6. Согласно методике проведения эксперимента, его итоговый результат также является массивом, i-ый элемент которого определяется на основе полученных массивов данных по следующей формуле: $a_{res}[i] = \#a_1[i] + \#a_2[i] + \#a_3[i] + \#a_4[i] + \#a_5[i] + \#a_6[i], i \in [1, 6]$, где на месте каждого символа # стоит один из двух коэффициентов, 1 или -1. Вася хотел нарисовать график получившегося результата, но что-то пошло не так, и он получил графики всех массивов данных, включая результирующий:



Открыть изображение можно в отдельной вкладке.

Известно, что ряды указаны по порядку в соответствии с номерами исходных массивов и среди них добавился результирующий, т.е. он может быть как в середине рядов, так и самым первым либо последним. Также известно, что сумма элементов результирующего массива данных больше суммы элементов любого из исходных массивов данных.

Помогите Васе найти среди всех графиков тот, который соответствует результирующему массиву. Для этого восстановите формулу, по которой получился график. В ответ запишите последовательность из знаков «+» и «-», где знак «+» означает, что в этой формуле вместо соответствующего символа «#» стоит коэффициент 1, а знак «-» – коэффициент -1. Например, для формулы $a_{res}[i] = (-1) \cdot a_1[i] + 1 \cdot a_2[i] + 1 \cdot a_3[i] + (-1) \cdot a_4[i]$ ответ будет выглядеть следующим образом: -++-.

Ответ: -++++-

3. Сортировка и фильтрация данных (3 балла)

[Похожие пользователи]

Сергей занимается анализом данных и в последнее время увлёкся сбором данных с одного известного видеохостинга. Итоговые данные сохраняются в несколько таблиц.

Таблица users хранит информацию о пользователях видеохостинга:

- id - уникальный идентификатор пользователя;
- name - фамилия и имя пользователя.

Таблица videos хранит информацию о хранимых видеозаписях:

- id - уникальный идентификатор видеозаписи;
- name - название видеозаписи;
- slug - краткий код видеозаписи.

Таблица views хранит информацию о просмотрах видеозаписей пользователями:

- user_id - идентификатор пользователя, просмотревшего видеозапись;
- video_id - идентификатор просмотренной видеозаписи.

Для первичного анализа Сергей выбрал пользователей с ID, равными 11 и 39. Он хочет изучить, какие видеозаписи были просмотрены и первым пользователем, и вторым. Определите это вместе с ним. В качестве ответа введите ID этих видеозаписей через пробел в порядке возрастания.

Для решения задачи Вы можете подключиться к базе данных напрямую или скачать её на компьютер в формате SQLite.

Примечание. Семантика операторов, которые могут понадобиться при решении задачи:

Оператор	Описание
SELECT ###	Выводит значения атрибутов или функций, вычисленных от множества значений атрибутов, перечисленных после ключевого слова SELECT
FROM ###	Указывает название таблицы, из которой берутся значения
JOIN table ON ###	Соединяет таблицу table с имеющейся по условию, указанному вместо ###
WHERE ###	Фильтр. В вывод SELECT, в том числе в качестве аргументов вычислимых функций, будут попадать только записи, удовлетворяющие условиям фильтра
GROUP BY ###	Группирует записи так, что в одной группе оказываются все записи с одинаковыми значениями атрибута ###. В этом случае функция, указанная в SELECT, вычисляется отдельно для каждой группы
COUNT(###)	Функция, вычисляющая количество значений атрибута ### для всех записей с учетом фильтра и группировки, если они используются
ORDER BY ###	Сортирует выводимые результаты по возрастанию (для атрибутов, являющихся строками – в лексикографическом порядке), в том числе по атрибуту, который сам не выводится оператором SELECT

Пример запроса:

```
SELECT Album.Title, COUNT(*) FROM Track
JOIN Album ON Album.AlbumId = Track.AlbumId
Where Track.Milliseconds >= 10*60*1000
GROUP BY Album.AlbumId
ORDER BY Album.Title;
```

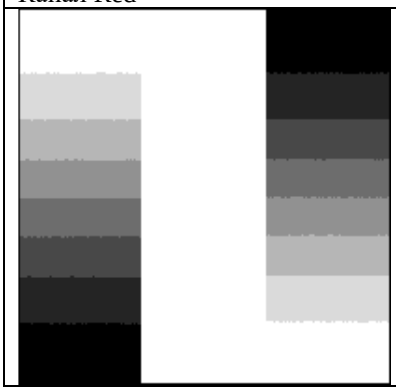
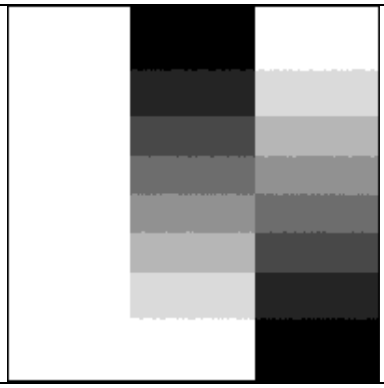
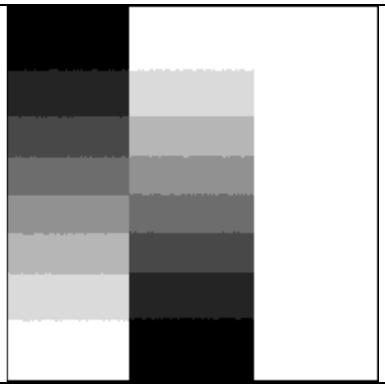
Данный запрос выбирает записи из таблицы `Track`, у которых значение поля `Milliseconds` больше или равно 600000. Далее с этой выборкой соединяется таблица `Album`, причём соединяются только те строки, где поле `AlbumId` в строке таблицы `Track` совпадает с полем `AlbumId` в таблице `Album`. Далее выполняется группировка результата соединения по полю `AlbumId` таблицы `Album`. После этого для каждой группы строк выводится значение поля `Title` таблицы `Album` и общее количество строк в данной группе. Последним действием полученная информация сортируется по полю `Title` таблицы `Album` по возрастанию.

Ответ: 16 26 47 61 62

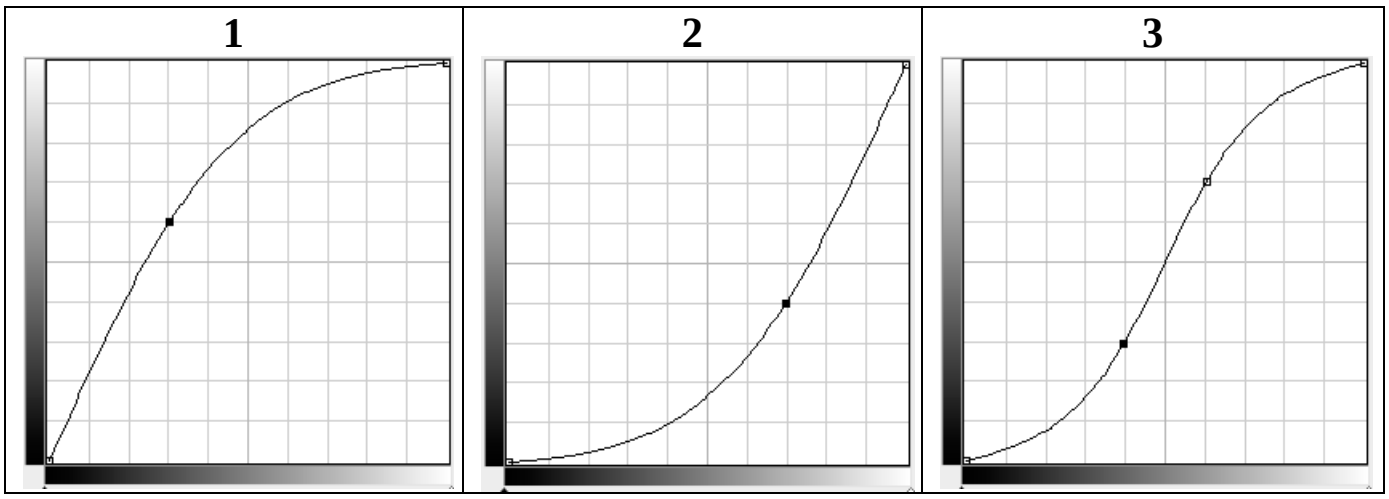
4. Мультимедиа технологии (2 балла)

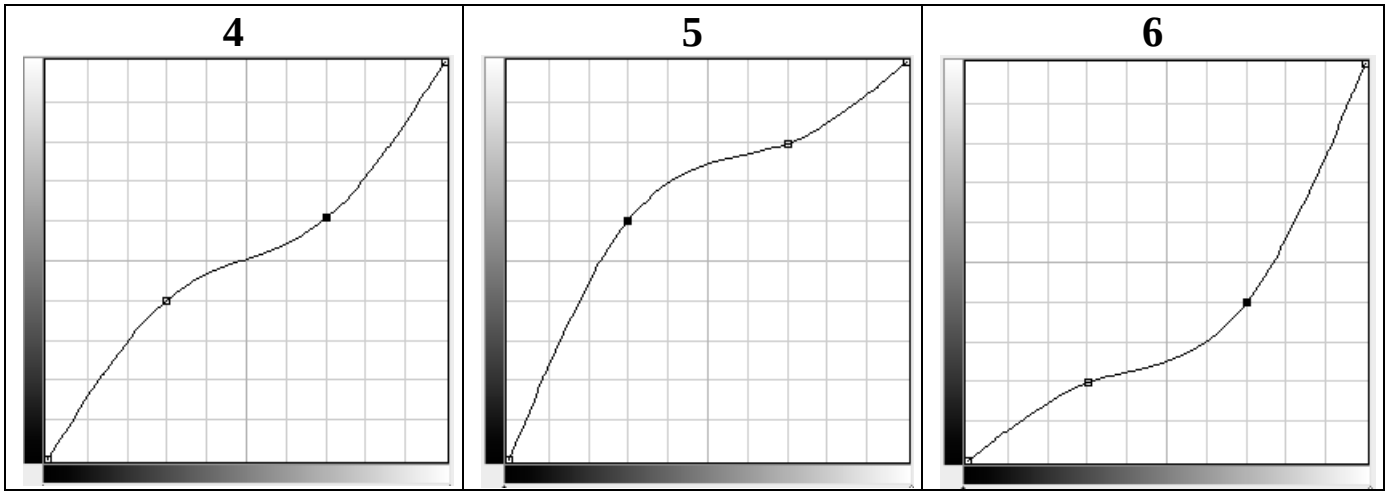
[Кривые и гистограммы]

Цветное изображение, кодированное в модели RGB, представлено картами яркостей по каждому из цветовых каналов.

Канал Red	Канал Green	Канал Blue
		

Одним из инструментов цветокоррекции изображения является Кривая (Curves). Этот инструмент может применяться, в том числе, к отдельным цветовым каналам. Кривая определяет, какому значению входной яркости (они отложены по оси абсцисс) соответствует какое значение выходной яркости (они отложены по оси ординат) после коррекции.





Была проведена цветокоррекция, в процессе которой к каждому каналу применили ровно одну из перечисленных кривых, причем каждая кривая была применена не более одного раза.

Затем для каждого канала построили гистограмму. На гистограмме по оси абсцисс гистограммы расположены возможные значения яркости этого канала слева направо от темных к светлым, а по оси ординат – количество пикселей изображения, имеющих такую яркость в соответствующем канале. Получился такой результат:

Канал Red	Канал Green	Канал Blue
		

Определите, к какому каналу была применена какая кривая. В ответе укажите через пробел три числа от 1 до 6 – номера примененных кривых в порядке следования цветовых каналов (сначала для Red, потом для Green, потом для Blue).
Пример записи ответа 1 2 3
Ответ: 235

5. Телекоммуникационные технологии (2 балла)

[FTP-геймер]

Для передачи файлов в интернете можно использовать протокол FTP (File Transfer Protocol). Это клиент-серверный протокол, в котором команды и данные передаются по двум разным каналам, управляющему каналу и каналу данных соответственно.

Рассмотрим некоторые команды протокола FTP.

По команде RETR Path сервер по каналу данных отправляет клиенту файл по указанному пути, после чего клиент должен сохранить файл на устройство.

По команде MLSD Path сервер по каналу данных отправляет клиенту информацию о содержимом указанного каталога (или информацию о файле, если по пути лежит файл). Информация о каждом элементе каталога возвращается в виде набора **фактов**. Каждый факт представляется в виде пары ключ-значение в формате Name=Value;.

Ниже представлены некоторые стандартные факты:

- Create — время создания в формате YYYYMMDDhhmmss (например, время создания 05 ноября 1995 10:17:54 будет записано как 19951105101754);
- Modify — время последнего изменения в формате YYYYMMDDhhmmss;
- Type — тип элемента каталога:
  - file — обычный файл;
  - dir — каталог;
  - cdir — текущий каталог;
  - pdir — родительский каталог;
- Size — размер файла в байтах, указывается только для файлов.

При этом не все факты должны присутствовать в ответе команды.

После набора фактов указывается название файла или папки, отделённое ровно одним пробелом.

Все элементы каталога описываются на отдельных строках, при этом для переноса строки используется два символа (CL RF). Перенос строки имеется также после последней строки вывода команды.

Команда возвращает одну строку с типом элемента каталога, равным cdir, в таком случае название элемента каталога равно полному пути к каталогу. Команда возвращает одну строку с типом элемента каталога, равным pdir, и тогда название элемента каталога указывается как две точки (..).

Пример выполнения команды MLSD tmp:

```
Type=cdir;Modify=19981107085215; /tmp
Type=pdir;Modify=19990112030508; ..
Type=file;Size=25730;Modify=19940728095854; capmux.tar.z
Type=file;Size=1830;Modify=19940916055648; hatch.c
Type=file;Size=25624;Modify=19951003165342; MacIP-02.txt
Type=file;Size=2154;Modify=19950501105033; uar.netbsd.patch
Type=file;Size=54757;Modify=19951105101754; iptnnladev.1.0.sit.hqx
Type=file;Size=226546;Modify=19970515023901; melbcs.tif
Type=file;Size=12927;Modify=19961025135602; tardis.1.6.sit.hqx
Type=file;Size=17867;Modify=19961025135602; timelord.1.4.sit.hqx
Type=file;Size=224907;Modify=19980615100045; uar.1.2.3.sit.hqx
Type=file;Size=1024990;Modify=19980130010322; cap60.pl198.tar.gz
```

Миша нашёл в интернете общедоступный FTP-сервер и начал исследовать его содержимое. Ниже представлено содержимое каталога /Games этого сервера:

Полный путь	**Размер (в байтах)**	**Тип**	**Время создания**
/	-	Каталог	12 января 2012 07:47:00
/Games	-	Каталог	10 марта 2014 10:55:00
/Games/0ad_0.0.22_amd64.appimage	1 030 321 584	Файл	29 мая 2018 14:00:00
/Games/battle-for-wesnoth-1.18.3-win.zip	506 219 599	Файл	28 октября 2024 20:45:00
/Games/endless-sky-v0.10.10.dmg	282 303 434	Файл	26 октября 2024 22:38:00
/Games/freeciv-gtk3.22-3.1.4-x86_64.appimage	155 636 928	Файл	03 января 2025 00:33:00
/Games/openttd_1.10.3_amd64.appimage	69 316 648	Файл	27 января 2021 14:41:00
/Games/super-tux-kart_1.2_amd64.appimage	594 681 896	Файл	22 февраля 2021 14:48:00
/Games/super-tux-v0.6.3-win64.msi	178 823 737	Файл	23 декабря 2021 01:25:00
/Games/unknown-horizons-2019.1.214-Setup-VC15-x86.exe	192 179 770	Файл	24 февраля 2019 16:23:00

Миша выполнил команду MLSD /Games, после чего выполнил несколько команд RETR. Известно, что всего по каналу данных было передано 575 536 968 байт и что команда MLSD на этом сервере возвращает факты Size, Type и Create. Выберите файлы, которые были получены с помощью команды RETR. Если есть несколько вариантов того, какие файлы могли быть получены, выберите тот, в котором меньше всего файлов.

Варианты ответа:

- 0ad_0.0.22_amd64.appimage
- battle-for-wesnoth-1.18.3-win.zip
- endless-sky-v0.10.10.dmg
- freeciv-gtk3.22-3.1.4-x86_64.appimage
- openttd_1.10.3_amd64.appimage
- super-tux-kart_1.2_amd64.appimage
- super-tux-v0.6.3-win64.msi
- unknown-horizons-2019.1.214-Setup-VC15-x86.exe
- Такого набора файлов не существует

Ответ: battle-for-wesnoth-1.18.3-win.zip || openttd_1.10.3_amd64.appimage

6. Операционные системы (3 балла)

[Активный процесс]

Вам дана виртуальная машина под управлением операционной системы Debian 12 с доступной учётной записью root. В этой виртуальной машине запущено несколько фоновых процессов, часть из которых через некоторые (возможно, разные для разных процессов, но одинаковые для отдельного процесса) промежутки времени выполняют запись в файл /root/output.txt. Определите PID процесса, который за 2 минуты сохранит в этот файл больше данных, чем любой другой процесс и сохраните полученное значение в файле /root/answer.txt. В файле не должно быть никакой другой информации, включая перевод строки.

7. Технологии программирования (4 балла)

[UNO!]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	4 секунды
Ограничение по памяти	256 мегабайт

Описанные ниже правила немного отличаются от стандартных правил UNO. Читайте условие внимательно!

Вам предстоит проэмулировать известную настольную игру UNO в рамках проекта по портированию игры на телефон. Для этого от вас требуется написать программу, которая по транскрипту действий игроков проверит, корректны ли эти ходы, и, если нет, выведет номер первого хода, сделанного не по правилам.

Колода для игры в UNO состоит из:

- обычных карт номиналом от 0 до 9 каждого из четырех цветов: красного ('R'), синего ('B'), зеленого ('G') и желтого ('Y');
- карт «сменить направление» ('@') и «взять две карты» ('+') каждого цвета;
- бесцветных карт «взять четыре карты» ('*').

В рамках данной задачи будем считать колоду бесконечной и содержащей неограниченное количество карт каждого типа.

Всего играют p игроков, у каждого изначально по a карт, одна карта лежит на столе, а остальные — в колоде рядом с ней. Первый ход делает первый игрок, направление перехода хода изначально — по увеличению номера игрока (то есть следующий ход делает второй игрок, затем третий, и так далее).

В свой ход игрок должен сыграть карту, совпадающую с последней выложенной картой либо по значению, либо по цвету. Бесцветная карта '*' считается за любой цвет: ее можно положить поверх любого цвета, и поверх нее можно положить любой цвет. Если игроку нечем ходить или он не хочет выкладывать карту, он берет карту из колоды и пропускает свой ход.

Стандартные правила игры меняются в следующих случаях.

- Если игрок выложил карту '@', направление перехода хода меняется на противоположное.
- Если игрок выложил карту '+', следующие за ним игроки должны также выкладывать карту '+' любого цвета, пока ход не дойдет до игрока, который не сможет или не захочет сыграть такую карту. Этот игрок возьмет $2x$ карт из колоды, где x — количество выложенных перед ним карт '+'; после чего ход перейдет к следующему игроку.
- Для карт '*' правило аналогично, но игрок, на котором остановится последовательность таких карт, возьмет из колоды $4x$ карт. Ход также перейдет к следующему игроку.

Также есть особое правило, называемое «перехват хода».

- Если у игрока есть карта, в точности (и по значению, и по цвету) совпадающая с последней выложенной, он может выложить ее вне своего хода, после чего ход перейдет к следующему относительно него игроку в текущем направлении.
- Перехватывать можно и карты действий, но при перехвате карт '+' и '*' число накопленных в цепочке карт x сбрасывается до 1.
- Если игрок в порядке стандартной очередности ходов выкладывает карту, совпадающую с верхней картой на столе, это не считается перехватом. То есть если первый игрок положил карту 'Y+', то ход третьего игрока картой 'Y+' считается перехватом, а ход второго игрока такой же картой — нет.
- Перехватить ход можно даже после того, как кто-то уже взял карту. Например, первый игрок походил, второй взял карту из колоды, и вместо хода третьего первый в качестве перехвата хода может выложить такую же карту, как и ход назад.

Финальное правило: игрок должен крикнуть «Uno!», когда у него на руке остается одна карта. Если игрок выкладывает на стол свою последнюю карту, он выходит из игры, и больше в ней не участвует.

По изначальному набору карт на руке у каждого игрока, изначальной карте на столе и последовательности ходов игроков, определите, не нарушал ли кто-то правила игры, и если нарушал — выведите первое такое нарушение.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода находится одно целое число t — количество наборов входных данных ($1 \le t \le 400$). Далее следует описание наборов входных данных.

В первой строке набора входных данных даны три целых числа n, a и q — количество игроков, изначальное число карт на руке и количество ходов, а также исходная карта на столе c0 ($1 \le n \le 10$; $1 \le a \le 10$; $1 \le q \le 500$). Все карты, кроме карты '*', задаются двумя символами, где первый задает цвет, а второй — значение карты (например, «R+» или «Y7»); карта '*' задается одним символом '*'. Изначальная карта имеет значение от '0' до '9', то есть не является картой действия.

Затем следуют n строк по a карт в каждой — i-я строка содержит изначальные карты на руке у i-го игрока. Карты задаются в формате, описанном выше.

Далее следует описание ходов. Каждый ход описывается последовательностью действий какого-то игрока в формате

- «i plays c» — i-й игрок положил на стол карту c (либо в свой ход, либо как перехват хода);
- «i says uno after he plays cc» — i-й игрок положил карту c и сказал «Uno!».
- «i takes k $c1$ $c2$ … ck» — i-й игрок взял k карт $c1$,$c2$,…,ck.

Для вашего удобства наборы входных данных в примере ниже разделены друг от друга пустыми строками. В реальных тестах этих пустых строк нет!

Формат выходных данных

Для каждого набора входных данных выведите вердикт по игре. Если за игру не было нарушений правил, выведите «ok», вне зависимости от того, сколько игроков осталось в игре. В рамках данной задачи будем считать, что даже когда в игре остается один игрок, он может продолжать делать ходы.

Если же кто-то из игроков нарушил правила, выведите первое нарушение в формате «*msg*, player *p*, turn *t*», где t ($1 \le t \le q$) — номер хода, на котором случилось нарушение, p ($1 \le p \le n$) — номер игрока, ходившего в этот ход, а *msg* — одно из следующих сообщений:

- «missing card», если игрок выкладывает карту, которой нет у него на руке;
- «wrong turn order», если игрок походил не в свою очередь (включая случай, когда уже выбывший игрок пытается походить);
- «wrong action», если игрок совершает неверное действие (кладет карту, когда должен взять карты; берет неверное число карт);
- «card doesn't match», если игрок выкладывает неподходящую карту;
- «fail 'Uno!'», если у игрока осталась одна карта, но он не сказал «Uno!», либо наоборот, осталось любое другое число карт, а «Uno!» было сказано.

Когда нарушение подходит под несколько из описанных форматов, используйте тот формат, который выше в списке. Например:

- игрок выложил неподходящую карту, которой не было у него на руке — нарушение «missing card», хотя «card doesn't match» тоже подошло бы, но оно ниже в списке;
- игрок выложил карту, которой не было у него на руке, когда должен был брать карты — «missing card», а не «wrong action»;
- игрок попытался перехватить ход, но выложил не ту карту с руки — «wrong turn order», а не «wrong action»;
- под игрока сыграли 'R+', и он должен был либо взять две карты, либо выложить еще одну '+', но выложил с руки 'Y2' — «wrong action», а не «card doesn't match».

Пример

Стандартный ввод	**Стандартный вывод**
3 2 4 9 R8 R+ R1 R3 Y5 R5 R5 R5 * 1 plays R+ 2 takes 2 G1 Y3 1 plays R1 2 plays R5 2 plays R5 2 plays R5 1 plays R3 2 plays Y3 1 plays Y5 3 3 11 B2 R@ Y@ B+ G@ B@ G+ B+ Y@ Y0 1 plays B+ 3 plays B+ 1 takes 2 Y1 Y2 2 plays B@ 1 plays R@ 2 says uno after he plays G@	fail 'Uno!', player 1, turn 7 wrong turn order, player 1, turn 9 missing card, player 2, turn 6

Стандартный ввод	Стандартный вывод
1 plays Y@ 3 says uno after he plays Y@ 1 plays Y1 2 takes 1 G+ 3 plays Y0 3 2 6 R7 * * * R+ R@ * 1 says uno after he plays * 1 plays * 2 says uno after he plays * 3 says uno after he plays * 2 takes 12 R+ B+ G+ Y+ R@ B0 G7 Y2 R1 * B2 B3 2 plays R9	

Замечание

В первой игре из примера:

1. первый игрок играет «взять две», и второй игрок берет две карты и пропускает ход;
2. первый игрок играет красную '1';
3. второй играет красную '5', после чего дважды перехватывает ход такой же картой;
4. первый игрок играет карту, но не говорит «Uno!», хотя после у него остается одна карта на руке.

Во второй игре из примера:

1. первый игрок играет «взять две»;
2. третий игрок перехватывает, ход переходит к первому, но количество карт '+' сбрасывается до 1 из-за перехвата, поэтому первый берет только две карты вместо четырех;
3. следующие четыре хода первый и второй каждым ходом меняют направление игры, поэтому ход от первого переходит второму, а от второго, после смены направления — первому;
4. после первого должен снова ходить второй (положительное направление хода), но третий перехватывает ход желтой картой «смена направления»; направление становится отрицательным, следующим должен ходить второй;
5. вместо этого ходит первый игрок — это нарушение правил.

В третьей игре из примера:

1. первыми двумя ходами первый играет '*' и сразу же перехватывает свой ход, выходя из игры;
2. второй и третий продолжают последовательность '**';
3. первый вышел из игры, поэтому следующий после третьего — это второй; ему приходится взять 12 карт (самая первая '*' не действует, так как была выложена до перехвата);
4. второй не в свой ход выкладывает карту (так как должен был пропустить ход), но так как такой карты у него вообще не должно быть, нарушение правил — «missing card».

8. Технологии программирования (3 балла) [Бактериальное окружение]

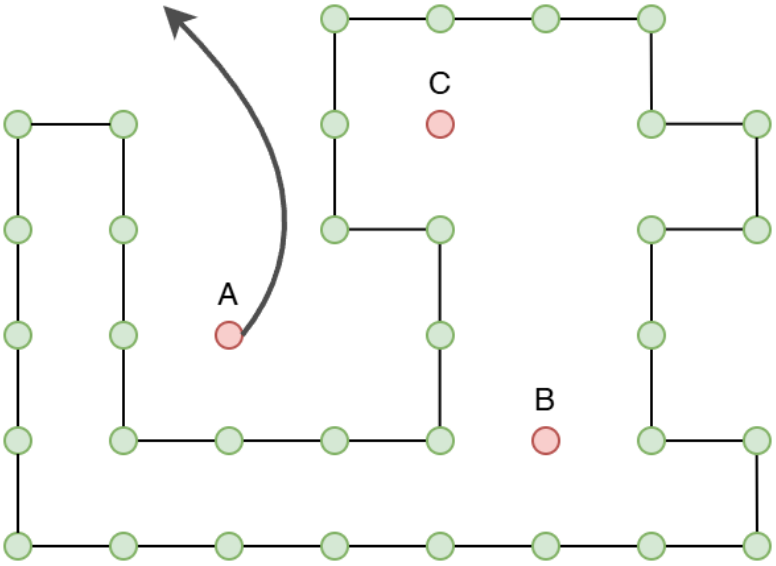
Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

В лаборатории проводят эксперименты с бактериями в чашке Петри. Чашку Петри можно представить как часть координатной плоскости размера $n \times m$ (координаты от 0 до n включительно по оси Ox и от 0 до m включительно по Oy).

В целочисленных точках чашки Петри можно размещать бактерии и наблюдать за их поведением. Будем называть две целочисленные точки $(x1,y1)$ и $(x2,y2)$ клетки *соседними*, если они находятся на расстоянии ровно 1, или же, что эквивалентно для целочисленных точек, если $|x2-x1|+|y2-y1|=1$.

Рассмотрим последовательность клеток $A1, \ldots, Ak$, в которой Ai — соседняя с $Ai \bmod k+1$ для всех $1 \le i \le k$. Тогда назовем *замкнутым контуром* замкнутую ломаную, проведенную по этим точкам в таком порядке. Также скажем, что другая точка находится *внутри* контура, если от нее до границы чашки не существует неразрывной линии на плоскости, не пересекающей этот контур.

Соответственно, на иллюстрации ниже виден контур из зеленых точек, и точка A находится вне контура, а B и C — внутри.



Изначально чашка Петри пустая. В рамках эксперимента в нее помещаются бактерии двух враждебно настроенных видов: красные и зеленые. Эксперимент проводят ученые, которые q раз добавляют по бактерии на поле по следующим правилам:

- новая бактерия всегда помещается в еще не занятую точку чашки, расположенную снаружи от всех имеющихся в чашке замкнутых контуров;
- если в какой-то момент эксперимента бактерии одного цвета образуют замкнутый контур, все бактерии внутри этого контура оказываются «захвачены» и меняют свой цвет на цвет контура.

К сожалению, чашка разбилась и все бактерии погибли, но у вас остался каталог со всеми q действиями, которые выполняли ученые. От вас требуется проанализировать эти действия и после каждого из них вывести число красных и зеленых бактерий в чашке.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода находится одно целое число t — количество наборов входных данных ($1 \le t \le 20$). Далее следует описание наборов входных данных.

В первой строке каждого набора данных даны три целых числа n, m и q — размеры чашки и количество действий ($0 \le n,m \le 100$; $1 \le q \le (n+1)(m+1)$).

В следующих qq строках даны по три числа ci, xi, yi — цвет и положение очередной бактерии ($1 \le ci \le 2$; $0 \le xi \le n$; $0 \le yi \le m$). Здесь ci=1 означает красную бактерию, а ci=2 — зеленую.

Гарантируется, что все запросы корректны по условию задачи.

Также гарантируется, что сумма $(n+1)\cdot(m+1)$ по всем наборам входных данных не превосходит 700.

Для вашего удобства наборы входных данных в примере ниже разделены друг от друга пустыми строками. В реальных тестах этих пустых строк нет!

Формат выходных данных

Для каждого добавления бактерии в чашку Петри в отдельной строке выведите через пробел два целых числа — количество красных и зеленых бактерий в чашке после этого действия.

В примере из условия пустые строки в выводе также добавлены ради удобства — каждый вывод напротив соответствующего действия. Ваше решение не должно выводить пустые строки.

Пример

Стандартный ввод	Стандартный вывод
3 1 1 4 1 0 0 2 0 1 2 1 0 1 1 1 5 2 18 1 4 1 2 1 1 1 0 0 2 5 2	 1 0 1 1 1 2 2 2 1 0 1 1 2 1 2 2

Стандартный ввод	Стандартный вывод
1 0 1 2 5 1 1 0 2 2 5 0 1 1 2 2 4 0 1 2 2 2 3 0 1 2 1 2 3 1 1 2 0 2 3 2 1 1 0 2 4 2 4 4 25 1 2 2 2 2 3 2 3 3 2 3 2 2 3 1 2 2 1 2 1 1 2 1 2 2 1 3 1 0 0 1 0 1 1 0 2 1 0 3 1 0 4 1 1 4 1 2 4 1 3 4 1 4 4 1 4 3 1 4 2 1 4 1 1 4 0 1 3 0 1 2 0 1 1 0	3 2 3 3 4 3 4 4 5 4 5 5 6 5 6 6 7 6 7 7 8 7 8 8 10 7 9 9 1 0 1 1 1 2 1 3 1 4 1 5 1 6 1 7 0 9 1 9 2 9 3 9 4 9 5 9 6 9 7 9 8 9 9 9 10 9 11 9 12 9 13 9 14 9 15 9 25 0

Задания для 9 и 10 класса

Заключительный этап, 10 класс (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балла)

[100 первых цифр]

Петя составляет ряд из периодических дробей, записанных в шестнадцатеричной системе счисления. Первый член ряда равен $0,(AB)_{16}$. Каждый следующий член ряда ровно в 4 раза меньше предыдущего. Известно, что у некоторого члена ряда сумма первых 100 цифр после запятой равна 441_{10}. Найдите и укажите в ответ номер этого члена ряда. Если такого члена ряда не существует, запишите в ответ NULL.

Ответ: 117

Решение:

Обратим внимание, что деление числа на 4 равносильно сдвигу запятой на один разряд влево в записи числа в четверичной системе счисления. Также вспомним, что мы можем перевести число из шестнадцатеричной системы счисления в четверичную заменив независимо каждый шестнадцатеричный разряд на два четверичных.

Тогда первый член ряда $\mathbf{0,(AB)_{16}} = 0,(2223)_4$.

$0,(2223)_4 / 4_{10} = 0,(2223)_4 / 10_4 = 0,0(2223)_4 = \mathbf{0,2(AE)_{16}}$ – это второй член ряда.

Повторим операцию деления несколько раз, зафиксировав закономерность в записи получающихся членах ряда.

$0,0(2223)_4 / 4_{10} = 0,0(2223)_4 / 10_4 = 0,00(2223)_4 = \mathbf{0,0(AB)_{16}}$ – третий член ряда.

$0,00(2223)_4 / 4_{10} = 0,00(2223)_4 / 10_4 = 0,000(2223)_4 = \mathbf{0,02(AE)_{16}}$ – четвертый член ряда.

Легко заметить, что все нечетные члены ряда будут получаться добавление одного нуля после запятой в предыдущий нечетный член ряда, а все четные – добавлением одного нуля после запятой в предыдущий четный член ряда.

Рассмотрим возможные суммы первых 100 цифр после запятой для нечетных членов ряда. Для первого члена ряда это будет 50*10+50*11 = 1050. Для третьего члена ряда 50*10+49*11 = 1039. То есть, у нечетных членов ряда сумма может быть представлена или как X*21 или как X*21-11, где X – натуральное число. Аналогично для четных членов ряда сумма будет равна или 2+Y*24-14 = Y*24-12 или 2+Y*24. Рассмотрим число из условия – 441_{10} и заметим, что это 21*21. Следовательно, это сумма цифр для нечетного члена ряда вида X*21, где X=21. Осталось вычислить его номер. Для первого члена ряда X=50, для пятого члена ряда X=49 и т.д. Тогда номер нужного члена рядам можно вычислить как (50-X)*4+1 = (50-21)*4+1=117

2. Кодирование информации. Объем информации (2 балла)

[Полёт]

Петя создает ПО для управления дроном. Дрон перемещается в трехмерном пространстве с заданной декартовой системой координат. Дрон умеет выполнять 6 возможных команд: вверх, вниз, вправо, влево, вперед и назад. Каждая команда перемещает дрон на 1 единицу длины в указанном направлении. Пространство не содержит никаких препятствий для перемещения дрона. Программа для дрона – это последовательность из определенного количества указанных команд. Пока дрон умеет выполнять только программы, состоящие ровно из 11 команд. Петя решил записывать программы для дрона в память в виде последовательности команд, кодируя каждую команду минимально возможным, одинаковым для всех команд количеством бит.

Вася заметил, что дрон всегда стартует из точки A и всегда заканчивает движение в точке B, смещенную на 2 единицы длины вправо, 3 единицы длины вверх и 4 единицы длины вперед. Вася сказал Пете, что тогда нет смысла выделять память исходя из необходимости хранить произвольную программу заданной длины. Он предложил перенумеровать натуральными числами все программы, которые смогут переместить дрона из точки A в точку B и сохранять в памяти номер такой программы, используя для каждого возможного номера минимально возможное, одинаковое для всех номеров количество бит. Насколько меньше бит потребуется Васе для хранения одной программы, чем предлагал Петя? В ответе укажите целое число.

Ответ:16

Решение:

Определим объём информации, который необходимо выделить для программы, записывая её способом Пети. Для кодирования шести команд будет достаточно $\lceil \log_2 \mathbf{6} \rceil = \mathbf{3}$ бит информации, соответственно, для 11 команд будет достаточно $\mathbf{3 \cdot 11 = 33}$ бит.

Определим объём информации, который необходимо выделить для программы, записывая её способом Васи. Для удобства обозначим команды буквами: вверх - U, вниз - D, вправо - R, влево - L, вперед - F, назад - B. Запишем минимальную программу, которая нужна для смещения на 2 единицы длины вправо, 3 единицы длины вверх и 4 единицы длины вперед: RRUUUFFFF. Эта программа состоит из 9 команд, но по условию в программе должно быть 11 команд, при этом итоговое перемещение дрона в пространстве должно остаться таким же. Для этого итоговая программа должна иметь вид RRUUUFFFFXY, где X и Y - две команды, выполняющие движение в противоположном направлении (либо U и D, либо R и L, либо F и B).

Посчитаем количество различных программ, которые могут быть сведены к RRUUUFFFFUD. Необходимо посчитать количество перестановок с повторениями, для этого можно либо воспользоваться готовой формулой, либо вывести формулу из тех рассуждений, что всего всего есть $\boldsymbol{P_{11} = 11!}$ перестановок 11 элементов, при этом перестановки, в которых меняются местами одинаковые буквы, мы не учитываем. Таких перестановок будет $\mathbf{2!} \cdot \mathbf{(3+1)!} \cdot \mathbf{4!} \cdot \mathbf{1!}$: две буквы R из минимальной программы, три буквы U из минимальной программы и ещё одна из пары UD, четыре буквы F из минимальной программы и одна буква D из пары UD. Тогда формула будет выглядеть следующим образом:

$$\frac{\mathbf{11!}}{\mathbf{2!} \cdot \mathbf{(3+1)!} \cdot \mathbf{4!} \cdot \mathbf{1!}} = \mathbf{34650}$$

Итого получилось 34650 программ.

Аналогичные рассуждения можно провести для программ RRUUUFFFRL и RRUUUFFFFB. Для первого случая количество программ равно:

$$\frac{\mathbf{11}!}{(\mathbf{2}+\mathbf{1})!\cdot\mathbf{3}!\cdot\mathbf{4}!\cdot\mathbf{1}!}=\mathbf{46200}$$

Для второго случая количество программ равно:

$$\frac{\mathbf{11}!}{\mathbf{2}!\cdot\mathbf{3}!\cdot(\mathbf{4}+\mathbf{1})!\cdot\mathbf{1}!}=\mathbf{27720}$$

И тогда на хранение номера одной программы нужно будет выделить $\lceil\log_2(\mathbf{34650}+\mathbf{46200}+\mathbf{27720})\rceil=\lceil\log_2\mathbf{108570}\rceil=\mathbf{17}$ бит.

Итоговый выигрыш по сравнению со способом Пети составит $\mathbf{33}-\mathbf{17}=\mathbf{16}$ бит.

3. Основы логики (3 балла)
[Кто такой Жегалкин?]

Есть логический преобразователь, на вход которому подается логическая функция от 20 переменных, не являющаяся тождественной ложью (то есть имеющая значение ИСТИНА хотя бы для одного набора значений переменных). Работает преобразователь следующим образом:

1. Строится таблица истинности данной функции.
2. Из полученной таблицы истинности берутся наборы переменных, для которых значение функции равно ИСТИНЕ.
3. Для каждого такого набора составляется полная конъюнкция всех 20 переменных, причем если значение этой переменной в наборе равно ЛЖИ, то в конъюнкции эта переменная будет с отрицанием, если значение переменной равно ИСТИНЕ, то в конъюнкции эта переменная будет без изменений.
4. Между всеми полученными полными конъюнкциями ставятся операции $\oplus$ (Исключающее ИЛИ).
5. Все переменные с отрицанием заменяются по следующему правилу:
   $\bar{X}=(X\oplus 1)$
6. Раскрываются все скобки по правилу: $X\wedge(Y\oplus Z)=X\wedge Y\oplus X\wedge Z$. После раскрытия скобок должно получиться выражение, которое может содержать некоторые отдельные переменные, конъюнкции переменных и константу ИСТИНА, соединенные Исключающим ИЛИ.
7. Из выражения убираются повторяющиеся конъюнкции. В результате у нас получается новая форма записи исходной функции. Гарантируется, что для каждой входной функции эта форма уникальна.
8. Производится проверка, остались ли в получившейся форме записи выражения операции конъюнкции. Если остались – преобразователь возвращает ЛОЖЬ, если нет – возвращает ИСТИНА.

Сколько существует таких функций от 20 переменных, которые можно подать на вход преобразователю и для которых он вернет ИСТИНУ? В ответе укажите целое число.

Ответ: 2097151

Решение:

Разберем работу алгоритма. Выполнив пункты 1-3 мы получаем набор полных конъюнкций, которые при сложении дают СДНФ исходной функции (то есть дизъюнкцию конъюнкций, состоящих из полного набора переменных, от которых зависит исходная функция). Только на шаге 4 вместо ожидаемых дизъюнкций полные конъюнкции объединяются операцией Исключающее ИЛИ. Следовательно получается запись, вида A $\oplus$ B $\oplus$ C…, где A, B, C…. – полные конъюнкции, в которых могут присутствовать инвертированные переменные.

Проанализируем, что будет, если все инвертированные переменные заменяются на (a $\oplus$ 1) и по указанным правилам раскрываются скобки. Если в конъюнкции присутствовала одна инвертированная переменная, то X(a $\oplus$ 1) = Xa $\oplus$ X. Если в конъюнкции присутствовали две инвертированные переменные, то
X(a $\oplus$ 1)(b $\oplus$ 1)= (Xa $\oplus$ X)(b $\oplus$ 1) = (Xa $\oplus$ X)b $\oplus$ 1(Xa $\oplus$ X) = Xab $\oplus$ Xb $\oplus$ Xa $\oplus$ X

И так далее, если посмотреть, можно заметить, что после раскрытия скобок мы получаем форму записи вида A $\oplus$ B $\oplus$ C $\oplus$ …, где A, B, C, … – конъюнкции, в которых уже не обязательно присутствуют все переменные, так же важно отметить, что в них не будет инвертированных переменных, и эта конъюнкция может быть из одной переменной (либо какая то переменная, от которой зависит исходная функция, либо константа 1). Так же после избавления от пар повторяющихся конъюнкций, понимаем, что A, B, C… будут уникальными конъюнкциями.

Посчитаем, сколько у нас может быть функций от 20 переменных. Помним, что тождественная ложь нам не подходит по условию задачи. Тогда функций получается: $2^{(2\wedge 20)}-1$

Посчитаем, сколько у нас всего может получиться форм записи, полученных после 7-го пункта алгоритма. В каждую конъюнкцию каждая переменная у нас может как входить, так и не входить (если в конъюнкцию не входит ни одна переменная – это соответствует случаю, когда конъюнкция представляет из себя константу 1). Следовательно, всего вариантов возможных конъюнкций 2^{20}. Далее каждая такая конъюнкция может как входить, так и не входить в конечную форму записи, при этом хотя бы одна конъюнкция в конечной форме должна быть => всего таких форм у нас $2^{(2\wedge 20)}-1$.

По условию задачи, мы знаем, что для каждой входной функции получается своя уникальная форма записи. Так же мы получили, что количество входных функций и количество возможных форм записи одинаковы, следовательно, у каждой входной функции есть единственная и уникальная форма записи, представленная в алгоритме. Тогда, нам достаточно посчитать те формы, в которых конъюнкции представляют из себя одну переменную: всего таких конъюнкций будет 20+1=21 (20 переменных от которых зависит исходная функция и константа 1). Каждую такую конъюнкцию мы можем как включать, так и не включать в запись, но хотя бы одна конъюнкция должна присутствовать => всего удовлетворяющих нас форм записи (а значит и функций) = $2^{(n+1)}-1=2^{21}-1$.

4. Кодирование информации. Алгоритмы обработки кодированной информации (1 балл) [Разбегайся!]

Пусть N – шестизначное десятичное число. Последовательность цифр R получается по следующему алгоритму:

1. 1000 раз повторяется первая (самая левая) цифра числа N.
2. Цикл для всех последующих цифр числа N, двигаясь слева направо:

   После каждого элемента последовательности, полученной на предыдущем шаге цикла вставить два раза подряд эту цифру

Например, если N=123456, то первые 10 элементов последовательности R, получившейся после завершения алгоритма будут:

1, 6, 6, 5, 6, 6, 5, 6, 6, 4

Известно, что в получившейся последовательности элемент с номером 209161 равен 7, элемент с номером 242758 равен 3, а элемент с номером 189890 равен 1. Нумерация элементов слева направо с 1. Определите максимальное число N, при котором это возможно, и запишите его в ответ. Если такое число невозможно, запишите в ответ NULL.

Ответ: 399791

Решение:

Пусть исходное число: 123456, то есть его цифры соответствуют их номерам считая слева направо от 1.

Рассмотрим начало последовательности после каждого шага цикла:

122122122122…

133233233133…

Обратим внимание, что после каждого шага последовательность будет состоять из одинаковых подпоследовательностей, начинающихся с каждой из тысячи единиц. Также легко заметить, что длина такой подпоследовательности будет утраиваться на каждом шаге, а, значит, после вставки последней цифры (6) длина каждой подпоследовательности, начинающейся с 1 будет 3^5=243.

Рассмотрим элемент результирующей последовательности с номером 209161. Поделим номер на 243 с остатком. Остаток будет равен 181. То есть, это 181-й элемент подпоследовательности, начинающейся с 1. Обратим внимание, что в такой подпоследовательности, полученной на определенном шаге цикла, на всех позициях, остаток от деления номера которых на 3 равен 0 или 2 стоят цифры, которые вставлены на этом шаге. Но остаток от деления 181 на 3 равен 1. Значит, это не может быть последняя цифра исходного числа.

При переходе на следующий шаг цикла, номер элемента, который был в последовательности на предыдущем шаге (n), будет давать новый номер n*3-2. Поэтому символ на 181 позиции в предыдущей подпоследовательности стоял на позиции (181+2)/3=61. Остаток от деления 61 на 3 равен опять 1. Значит это не цифра, стоящая на пятой позиции в исходном числе. Символ на 61 позиции в предыдущей подпоследовательности стоял на позиции (61+2)/3=21. 21 нацело делится на 3. Следовательно, мы знаем, что это цифра, которая в исходном числе стоит на позиции 4. Таким образом, мы определили, что в исходном числе на позиции 4 стоит цифра 7.

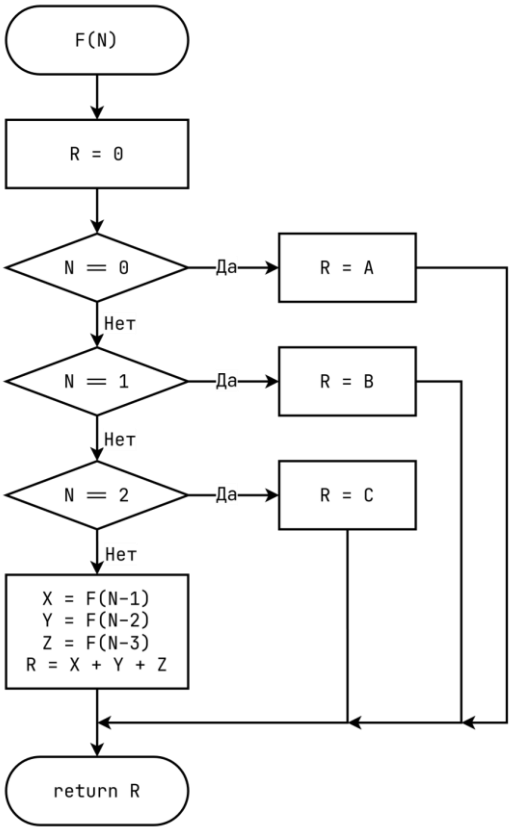
Аналогичными рассуждениями мы получим, что на позиции 1 в исходном числе стоит цифра 3 (остаток от деления 242758 на 243 равен 1), а на позиции 6 стоит цифра 1 (остаток от деления 189890 на 243 равен 107, а остаток от деления 107 на 3 равен 2). Поскольку нас просят найти максимальное число, то остальные его цифры должны быть максимальными, то есть равны 9.

Значит, это число 399791.

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (1 балл)

[Непизанские кролики]

Дана блок-схема алгоритма, реализованного в виде рекурсивно вызываемой функции:



Известно, что A, B и C - натуральные числа и что $A < B < C$.

Петя вызывает эту функцию, передавая ей в качестве входного параметра натуральное число. Известно, что для некоторого k функция возвращает следующие значения:

$F(k) = 50890368413$;

$F(k+1) = 93601980590$;

$F(k+2) = 172160883161$.

Определите значения A, B и C, при которых это возможно. Если таких вариантов несколько, выберите вариант с наименьшей суммой A, B и C. В ответе введите в указанном порядке значения A, B и C, разделённые пробелом.

Ответ: 1 2 5

Решение:

На блок-схеме изображён модифицированный алгоритм для генерации N-го числа Фибоначчи: в нём даются первые три числа, а следующие числа определяются как сумма трёх предыдущих (при A=0, B=0 и C=1 такая последовательность называется числами трибоначчи). Соответственно, нужно найти такие три числа, которые сгенерируют ряд с указанными в условии элементами. Для этого можно написать программу, которая принимает числа из условия на вход и вычитает первые два из третьего, тем самым генерируя предыдущие элементы последовательности:

```
d, e, f = 50890368413, 93601980590, 172160883161

while d > 0:
    t = f - d - e
    print(t)
    d, e, f = t, d, e
```

Результат работы программы нужно изучить достаточно внимательно, ведь в конце есть лишние элементы, которые не входят в последовательность, поскольку должно соблюдаться, что A,B,C∈N и A<B<C. В данном варианте окончание вывода будет таким:

...
15
8
5
2
1
2
-1

Две последние строки нас не интересуют, и в итоге A=1,B=2,C=5.

6. Телекоммуникационные технологии (2 балла).
[BGP]

Интернет состоит из отдельных крупных сетей, управляемых отдельными организациями. Между такими сетями распределяется пространство IP-адресов и устанавливается техническое и организационное взаимодействие. Несколько упрощая, можно сказать, что такие крупные сети называются автономными системами (AS). За каждой AS закрепляется IP-подсеть.

Для обмена маршрутной информацией между маршрутизаторами AS используется внешний протокол динамической маршрутизации BGP (Border Gateway Protocol). BGP выбирает наилучший маршрут для передачи пакетов на основе ряда атрибутов и метрик.

BGP-роутер при маршрутизации пакета выбирает IP-адрес следующего по маршруту маршрутизатора, который в BGP называется «соседом» (neighbor), по набору атрибутов маршрутов. В задаче для упрощения будем считать, что реализуется следующий алгоритм.

1. по адресу назначения в IP-пакете выбираются подходящие маршруты из таблицы маршрутизации по полю «сеть назначения».
2. потом из них выбирается лучший, для чего **последовательно** проверяются атрибуты маршрутов. То есть сначала проверяется первый атрибут всех подходящих маршрутов, и если у возможных маршрутов он имеет равное значение, проверяется второй атрибут и т.д.

В задаче используются следующие атрибуты в указанном порядке:

1. Локальное предпочтение (Local Preference)

Local Preference — это атрибут, который используется для выбора исходящего трафика из автономной системы. Маршрут с **наибольшим** значением Local Preference считается предпочтительным.

2. Наименьший Путь AS (AS Path)

BGP предпочитает маршруты с **наименьшим** количеством автономных систем в пути (AS Path). Чем короче путь, тем лучше.

3. Наименьший IGP Metric до Next Hop

Если есть несколько маршрутов с одинаковыми атрибутами, BGP выбирает маршрут с **наименьшим** значением метрики IGP до Next Hop (следующего прыжка).

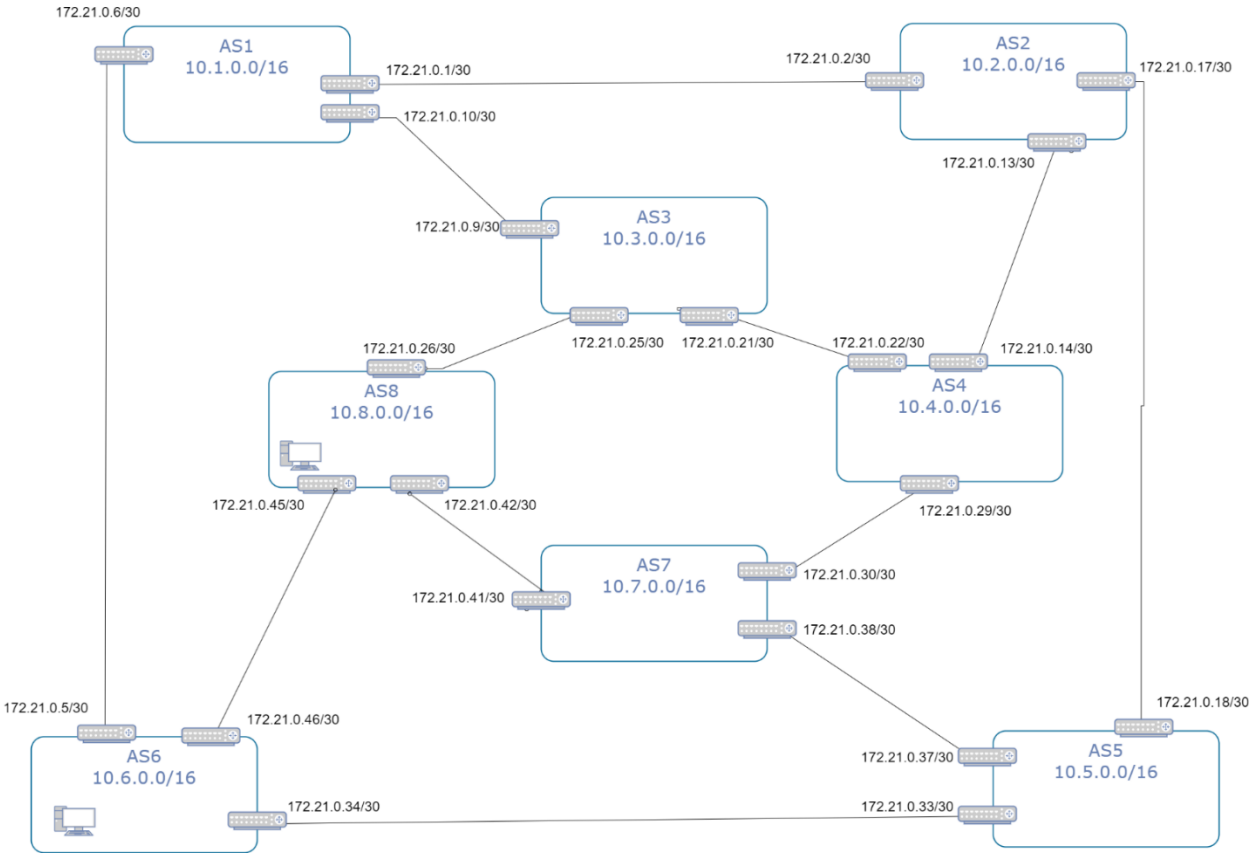
4. Старейший маршрут (Oldest Route)

Если все предыдущие атрибуты равны, BGP предпочитает **старейший** маршрут, так как он считается более стабильным.

5. Наименьший IP-адрес соседа

В крайнем случае, если все предыдущие атрибуты равны, BGP выбирает маршрут, полученный от соседа с **наименьшим** IP-адресом (сравнение производится как сравнение 32-битных значений адресов).

Далее приведены схема сети и фрагменты таблиц маршрутов для каждой из AS (сделано это для простоты, в реальности эти данные есть в каждом граничном маршрутизаторе AS).



AS1:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.2
10.1.0.0/16	Локально	500	0	0	2025-02-01	-
10.3.0.0/16	AS3	200	1	10	2025-02-03	172.21.0.9
10.7.0.0/16	AS2	100	3	20	2025-02-03	172.21.0.2
10.8.0.0/16	AS3	100	2	20	2025-02-01	172.21.0.9
10.6.0.0/16	AS6	100	2	20	2025-02-01	172.21.0.5
10.6.0.0/16	AS3	100	2	20	2025-02-01	172.21.0.9
10.3.0.0/16	AS6	200	3	50	2025-02-23	172.21.0.5
10.4.0.0/16	AS3	200	2	5	2025-02-03	172.21.0.9

AS2:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.2.0.0/16	Локально	500	0	0	2025-02-01	-
10.3.0.0/16	AS1	100	2	20	2025-02-03	172.21.0.1
10.6.0.0/16	AS1	100	2	20	2025-02-10	172.21.0.1
10.7.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.14
10.8.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.6.0.0/16	AS5	100	2	20	2025-02-03	172.21.0.18
10.3.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.4.0.0/16	AS1	200	2	20	2025-02-03	172.21.0.1
10.4.0.0/16	AS4	100	1	20	2025-02-03	172.21.0.14
10.1.0.0/16	AS1	100	1	20	2025-02-03	172.21.0.1
10.1.0.0/16	AS4	200	3	20	2025-02-03	172.21.0.14

AS3:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.22
10.6.0.0/16	AS1	100	2	20	2025-02-01	172.21.0.10
10.6.0.0/16	AS4	200	4	10	2025-02-03	172.21.0.22
10.7.0.0/16	AS4	100	3	10	2025-02-04	172.21.0.22
10.3.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS8	100	3	10	2025-02-04	172.21.0.26
10.4.0.0/16	AS4	200	1	10	2025-02-04	172.21.0.22
10.4.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS1	100	1	20	2025-02-04	172.21.0.10

AS4:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.13
10.6.0.0/16	AS2	100	3	5	2025-02-03	172.21.0.13
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	AS7	100	3	10	2025-02-01	172.21.0.30
10.8.0.0/16	AS3	100	1	10	2025-02-01	172.21.0.21
10.6.0.0/16	AS3	100	3	5	2025-02-11	172.21.0.21
10.1.0.0/16	AS3	100	3	5	2025-02-01	172.21.0.21

AS5:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.4.0.0/16	AS2	100	2	10	2025-02-02	172.21.0.17
10.5.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	AS6	100	1	20	2025-02-03	172.21.0.34
10.8.0.0/16	AS7	100	3	20	2025-02-01	172.21.0.38

10.3.0.0/16	AS2	500	3	20	2025-02-01	172.21.0.17
10.2.0.0/16	AS2	100	3	20	2025-02-01	172.21.0.17
10.3.0.0/16	AS6	100	3	20	2025-02-01	172.21.0.34
10.4.0.0/16	AS7	100	2	20	2025-02-02	172.21.0.38
10.1.0.0/16	AS6	100	2	20	2025-02-02	172.21.0.34
10.1.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.17

AS6:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS1	100	3	20	2025-02-02	172.21.0.6
10.6.0.0/16	Локально	500	0	0	2025-02-01	-
10.4.0.0/16	AS5	100	3	51	2025-02-11	172.21.0.33
10.7.0.0/16	AS1	100	4	20	2025-02-03	172.21.0.6
10.8.0.0/16	AS1	100	3	20	2025-02-01	172.21.0.6
10.3.0.0/16	AS8	200	2	5	2025-03-08	172.21.0.45
10.4.0.0/16	AS1	100	3	52	2025-02-11	172.21.0.6
10.3.0.0/16	AS5	100	4	50	2025-02-11	172.21.0.33
10.4.0.0/16	AS8	100	3	50	2025-02-11	172.21.0.45

AS7:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	3	20	2025-02-02	172.21.0.29
10.6.0.0/16	AS5	100	2	20	2025-02-03	172.21.0.37
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS4	100	3	20	2025-02-01	172.21.0.29
10.4.0.0/16	AS4	100	1	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS8	300	3	20	2025-02-01	172.21.0.42
10.1.0.0/16	AS4	200	3	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS5	400	5	80	2025-02-01	172.21.0.37

AS8:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.6.0.0/16	AS3	200	3	10	2025-02-01	172.21.0.25
10.7.0.0/16	AS3	200	3	10	2025-02-05	172.21.0.25
10.5.0.0/16	AS3	200	2	10	2025-02-03	172.21.0.25
10.4.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.8.0.0/16	Локально	500	0	0	2025-02-01	-
10.2.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.6.0.0/16	AS7	200	3	20	2025-02-01	172.21.0.41
10.3.0.0/16	AS3	200	1	10	2025-02-15	172.21.0.25
10.4.0.0/16	AS3	100	2	10	2025-02-03	172.21.0.25
10.6.0.0/16	AS6	50	1	20	2025-02-03	172.21.0.46

Определите сначала маршрут прохождения пакета из AS8 в AS6, а потом маршрут прохождения пакета в случае, если из-за аварии будет отключены связи между AS8 и AS3, а также между AS1 и AS6, а записи, соответствующие этим связям, будут исключены из таблиц.

В ответ запишите номера автономных систем через запятую, начиная с 8 и заканчивая 6, сначала для первого случая, а потом, через тире, для второго. Если маршрут построить нельзя (например, его нет или образуется цикл), вместо перечисления номеров автономных систем укажите NULL.

Пример ввода ответа: 8,10,12,6-8,10,1,2,6.

Ответ: 8,3,4,2,5,6-8,7,5,6

Решение:

В данной задаче необходимо аккуратно пройтись по таблицам маршрутизации и последовательно выбирать нужные строки.

Поскольку маршрут начинается в AS8, посмотрим таблицу маршрутизации этой автономной системы. В ней нужно выбрать те маршруты, в которых сеть назначения равна 10.6.0.0/16. Строки с такими маршрутами выделим зелёным цветом:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.6.0.0/16	**AS3**	**200**	**3**	**10**	**2025-02-01**	**172.21.0.25**
10.7.0.0/16	AS3	200	3	10	2025-02-05	172.21.0.25

10.5.0.0/16	AS3	200	2	10	2025-02-03	172.21.0.25
10.4.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.8.0.0/16	Локально	500	0	0	2025-02-01	-
10.2.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.6.0.0/16	AS7	200	3	20	2025-02-01	172.21.0.41
10.3.0.0/16	AS3	200	1	10	2025-02-15	172.21.0.25
10.4.0.0/16	AS3	100	2	10	2025-02-03	172.21.0.25
10.6.0.0/16	AS6	50	1	20	2025-02-03	172.21.0.46

При сравнении Local Preference из рассмотрения убирается маршрут в AS6, так как там значение меньше, а дальше при сравнении IGP Metric убирается маршрут в AS7, поскольку там значение больше. Остаётся маршрут в AS3.

Дальше аналогично рассмотрим маршруты из AS3:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.22
10.6.0.0/16	AS1	100	2	20	2025-02-01	172.21.0.10
10.6.0.0/16	**AS4**	**200**	**4**	**10**	**2025-02-03**	**172.21.0.22**
10.7.0.0/16	AS4	100	3	10	2025-02-04	172.21.0.22
10.3.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS8	100	3	10	2025-02-04	172.21.0.26
10.4.0.0/16	AS4	200	1	10	2025-02-04	172.21.0.22
10.4.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS1	100	1	20	2025-02-04	172.21.0.10

При сравнении Local Preference из рассмотрения убирается маршрут в AS1, и остаётся маршрут в AS4.

Рассмотрим маршруты из AS4:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.13
10.6.0.0/16	**AS2**	**100**	**3**	**5**	**2025-02-03**	**172.21.0.13**
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	AS7	100	3	10	2025-02-01	172.21.0.30
10.8.0.0/16	AS3	100	1	10	2025-02-01	172.21.0.21
10.6.0.0/16	AS3	100	3	5	2025-02-11	172.21.0.21
10.1.0.0/16	AS3	100	3	5	2025-02-01	172.21.0.21

При сравнении IGP Metric из рассмотрения убирается маршрут в AS7 (до этого все метрики равны), при сравнении даты появления маршрута из рассмотрения также убирается маршрут в AS3, так как он появился позже. Остаётся маршрут в AS2.

Рассмотрим маршруты из AS2:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.2.0.0/16	Локально	500	0	0	2025-02-01	-
10.3.0.0/16	AS1	100	2	20	2025-02-03	172.21.0.1
10.6.0.0/16	AS1	100	2	20	2025-02-10	172.21.0.1
10.7.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.14
10.8.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.6.0.0/16	**AS5**	**100**	**2**	**20**	**2025-02-03**	**172.21.0.18**
10.3.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.4.0.0/16	AS1	200	2	20	2025-02-03	172.21.0.1
10.4.0.0/16	AS4	100	1	20	2025-02-03	172.21.0.14
10.1.0.0/16	AS1	100	1	20	2025-02-03	172.21.0.1
10.1.0.0/16	AS4	200	3	20	2025-02-03	172.21.0.14

У двух нужных маршрутов все метрики до даты появления совпадают, а маршрут в AS5 старше маршрута в AS1, поэтому маршрут в AS1 убирается из рассмотрения, и остаётся маршрут в AS5.

Рассмотрим маршруты из AS5:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.4.0.0/16	AS2	100	2	10	2025-02-02	172.21.0.17
10.5.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	**AS6**	**100**	**1**	**20**	**2025-02-03**	**172.21.0.34**
10.8.0.0/16	AS7	100	3	20	2025-02-01	172.21.0.38
10.3.0.0/16	AS2	500	3	20	2025-02-01	172.21.0.17
10.2.0.0/16	AS2	100	3	20	2025-02-01	172.21.0.17

10.3.0.0/16	AS6	100	3	20	2025-02-01	172.21.0.34
10.4.0.0/16	AS7	100	2	20	2025-02-02	172.21.0.38
10.1.0.0/16	AS6	100	2	20	2025-02-02	172.21.0.34
10.1.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.17

В нём только один маршрут ведёт в нужную сеть, поэтому рассмотрим маршруты из AS6:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS1	100	3	20	2025-02-02	172.21.0.6
10.6.0.0/16	**Локально**	**500**	**0**	**0**	**2025-02-01**	**-**
10.4.0.0/16	AS5	100	3	51	2025-02-11	172.21.0.33
10.7.0.0/16	AS1	100	4	20	2025-02-03	172.21.0.6
10.8.0.0/16	AS1	100	3	20	2025-02-01	172.21.0.6
10.3.0.0/16	AS8	200	2	5	2025-03-08	172.21.0.45
10.4.0.0/16	AS1	100	3	52	2025-02-11	172.21.0.6
10.3.0.0/16	AS5	100	4	50	2025-02-11	172.21.0.33
10.4.0.0/16	AS8	100	3	50	2025-02-11	172.21.0.45

Здесь запрос в сеть назначения обрабатывается локально, т.е. текущей AS. Соответственно, пакет доставлен в AS6.

Итоговый маршрут получается следующим: **8,3,4,2,5,6**.

Дальше нужно так же пройтись по таблицам с учётом ситуации, когда из строя выходят каналы связи AS8 – AS3 и AS1 – AS6. Поскольку каналы связи двухсторонние, из таблиц маршрутизации удаляются записи не только из AS8 в AS3 и из AS1 в AS6, но и наоборот. Это сильно повлияет на маршрут пакета, поскольку исходно в первый раз пакет передаётся как раз из AS8 в AS3.

Рассмотрим маршруты из AS3 с учётом ограничений. Красным цветом выделены недоступные маршруты:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.6.0.0/16	AS3	200	3	10	2025-02-01	172.21.0.25
10.7.0.0/16	AS3	200	3	10	2025-02-05	172.21.0.25
10.5.0.0/16	AS3	200	2	10	2025-02-03	172.21.0.25
10.4.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.8.0.0/16	Локально	500	0	0	2025-02-01	-
10.2.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.6.0.0/16	**AS7**	**200**	**3**	**20**	**2025-02-01**	**172.21.0.41**
10.3.0.0/16	AS3	200	1	10	2025-02-15	172.21.0.25
10.4.0.0/16	AS3	100	2	10	2025-02-03	172.21.0.25
10.6.0.0/16	AS6	50	1	20	2025-02-03	172.21.0.46

Из двух оставшихся маршрутов, которые подходят, нужно выбрать маршрут в AS7, поскольку его значение Local Preference больше, чем у маршрута в AS6.

Рассмотрим маршруты из AS7:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	3	20	2025-02-02	172.21.0.29
10.6.0.0/16	**AS5**	**100**	**2**	**20**	**2025-02-03**	**172.21.0.37**
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS4	100	3	20	2025-02-01	172.21.0.29
10.4.0.0/16	AS4	100	1	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS8	300	3	20	2025-02-01	172.21.0.42
10.1.0.0/16	AS4	200	3	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS5	400	5	80	2025-02-01	172.21.0.37

Здесь подходит только один маршрут, в AS5. Рассмотрим маршруты из AS5:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.4.0.0/16	AS2	100	2	10	2025-02-02	172.21.0.17
10.5.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	**AS6**	**100**	**1**	**20**	**2025-02-03**	**172.21.0.34**
10.8.0.0/16	AS7	100	3	20	2025-02-01	172.21.0.38
10.3.0.0/16	AS2	500	3	20	2025-02-01	172.21.0.17
10.2.0.0/16	AS2	100	3	20	2025-02-01	172.21.0.17
10.3.0.0/16	AS6	100	3	20	2025-02-01	172.21.0.34
10.4.0.0/16	AS7	100	2	20	2025-02-02	172.21.0.38
10.1.0.0/16	AS6	100	2	20	2025-02-02	172.21.0.34
10.1.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.17

Таблица аналогична таблице в прошлом случае, поэтому рассмотрим маршруты из AS6:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS1	100	3	20	2025-02-02	172.21.0.6
10.6.0.0/16	**Локально**	**500**	**0**	**0**	**2025-02-01**	**-**
10.4.0.0/16	AS5	100	3	51	2025-02-11	172.21.0.33
10.7.0.0/16	AS1	100	4	20	2025-02-03	172.21.0.6
10.8.0.0/16	AS1	100	3	20	2025-02-01	172.21.0.6
10.3.0.0/16	AS8	200	2	5	2025-03-08	172.21.0.45
10.4.0.0/16	AS1	100	3	52	2025-02-11	172.21.0.6
10.3.0.0/16	AS5	100	4	50	2025-02-11	172.21.0.33
10.4.0.0/16	AS8	100	3	50	2025-02-11	172.21.0.45

Таблица, хоть и содержит удалённые маршруты, всё равно имеет только один маршрут для пакетов в нужную сеть, причём локальный, то есть пакет доставлен в AS6.

Итоговый маршрут получается следующим: **8,7,5,6**.

Ответ к задаче будет записываться следующим образом: **8,3,4,2,5,6-8,7,5,6**.

7. Технологии сортировки и фильтрации данных (2 балл) [Космический капитализм]

В свободное от работы время Николай Владимирович, ответственный за систему проведения олимпиад, разрабатывает новый космический симулятор, и на данный момент у него готова система инвентаря корабля с оборудованием, гиперпрыжки между двумя звёздными системами и торговля на планетах. Рассмотрим подробнее устройство симулятора.

Игровой мир состоит из звёздных систем, в каждой из которых есть несколько планет. Для перемещения корабля между звёздными системами необходим двигатель и топливный бак. На один гиперпрыжок корабль тратит количество топлива, равное расстоянию между звёздными системами в парсеках. Из этого следует, что корабль не может выполнить прыжок на расстояние больше, чем вместимость его топливного бака.

Владелец корабля имеет определённое количество галактических кредитов на счёте. На каждой планете представлены товары пяти видов: продовольствие, медикаменты, техника, роскошь и минералы. Для каждой планеты и каждого вида товара известно количество имеющегося товара в тоннах, стоимость покупки и продажи в галактических кредитах. Корпус корабля имеет заданную грузоподъёмность, и нельзя купить товаров больше, чем имеющаяся грузоподъёмность.

Сейчас Николай Владимирович пишет тест для проверки этой функциональности. Тестовый сценарий состоит из нескольких шагов:

1. Приобрести товар **одного вида** на текущей планете в некотором количестве;
2. Взлететь с планеты и выполнить гиперпрыжок **в другую систему**;
3. Приземлиться на выбранной планете и продать весь купленный на шаге 1 товар.

Для этого сценария есть отдельный файл сохранения, для которого известно количество товара на каждой планете, стоимость его покупки и продажи. Эти данные представлены в виде отдельной базы данных, которая доступна ниже.

Таблица stars хранит информацию о звёздных системах:

- id — уникальный идентификатор системы в рамках игрового мира;
- name — название системы.

Таблица planets хранит информацию о планетах:

- id — уникальный идентификатор планеты в рамках игрового мира;
- name — название планеты;
- star_id — идентификатор звёздной системы, к которой относится планета.

Таблица star_map хранит информацию о расстояниях между звёздными системами:

- star_from_id — идентификатор начальной звёздной системы;
- star_to_id — идентификатор звёздной системы назначения (обязательно отличается от star_from_id);
- distance — расстояние между этими звёздными системами в парсеках.

Таблица product_types хранит информацию о типах товаров:

- id — уникальный идентификатор типа товаров;
- name — название типа товаров.

Таблица prices хранит информацию о стоимости покупки и продажи каждого типа товаров на каждой из планет:

- planet_id — идентификатор планеты;
- product_type — идентификатор типа товаров;
- buy_price — стоимость покупки владельцем корабля одной тонны товара в галактических кредитах;
- sell_price — стоимость продажи владельцем корабля одной тонны товара в галактических кредитах;
- amount — количество товара в тоннах, доступное к покупке на данной планете.

В рамках данного сценария кредитов хватит для покупки любого количества любого товара на любой планете. Пока что в этом сценарии не определено, какой товар и в каком количестве нужно приобрести и на какой планете его продать. Николай Владимирович хочет подобрать эти параметры так, чтобы максимизировать полученную прибыль (разницу между стоимостью продажи и покупки). Известно, что в сохранении корабль находится на планете Раалито системы Хезе, грузоподъёмность корабля равна 80 тонн, а вместимости топливного бака хватит на прыжок расстоянием в 25 парсек. Как опытный программист, Николай Владимирович определил нужные параметры за несколько минут и дописал тест. Определите это вместе с ним и вы. В качестве ответа введите одно целое число — прибыль в галактических кредитах, полученную после выполнения тестового

сценария с найденными параметрами. Если окажется, что для тестового сценария нет ни одного набора параметров, который бы дал положительную прибыль, в качестве ответа введите 0.

Ответ: 6080

Решение:

Чтобы не писать длинный запрос с подзапросами, разобьём решение на три части.

Для начала нужно узнать идентификаторы планеты и соответствующей планетарной системы. Это можно выполнить одним запросом (заодно убедимся, что условие верное, и планета действительно находится в указанной системе):

```
SELECT p.id, p.name, s.id, s.name FROM planets p
INNER JOIN main.stars s ON s.id = p.star_id
WHERE p.name = 'Раалито';
```

Получится следующий вывод:

p.id	p.name	s.id	s.name
54	Раалито	10	Хезе

Теперь определим все планеты, которые находятся в системах, удалённых от текущей не более чем на 25 парсек:

```
SELECT p.* FROM star_map sm
INNER JOIN planets p ON star_to_id = p.star_id
WHERE star_from_id = 10 AND distance <= 25;
```

Получится следующая таблица:

id	name	star_id
1	Арнарик Гаудад	1
2	Рамгатру	1
3	Хартис	1
4	Проту Памперой	1
5	Калайна	1
6	Унашера	1
81	Гешши	17
82	Халгалла	17
83	Ципца	17
84	Эйя	17
85	Суджим	17
103	Оннд	22
104	Ушалла	22
105	Индерон	22
106	Энжа	22
107	Гаших	22
108	Уррагах	22
133	Зэмин	28
134	Рэдэа	28
135	Пунэ-анита	28
136	Хуку	28
167	Перегон	37
168	Глоткус	37
169	Мабэл	37
170	Дарзания	37
171	Ойкнур Бад	37
183	Альянони	40
184	Горик	40
185	Эгмер	40
186	Итниклей	40
299	Локон	66

300	Га-по	66
301	Иоханг	66
302	Шолоани	66
303	Пунисе	66

Третий запрос позволит построить таблицу с возможной прибылью от продажи того или иного товара на той или иной планете. Здесь необходимо вспомнить про ограничение грузоподъёмности корабля в 80 тонн:

```
SELECT pr.planet_id, pr.product_type, pr.sell_price, pr_from.buy_price,
pr_from.amount,
       MIN(pr_from.amount, 80) * (pr.sell_price - pr_from.buy_price) profit
FROM prices pr
INNER JOIN prices pr_from ON pr_from.planet_id = 54 AND pr_from.product_type =
pr.product_type
WHERE pr.planet_id IN (1, 2, 3, 4, 5, 6, 81, 82, 83, 84, 85, 103, 104, 105, 106, 107,
108, 133, 134, 135, 136, 167, 168, 169, 170, 171, 183, 184, 185, 186, 299, 300, 301,
302, 303)
ORDER BY profit DESC;
```

Первые 10 строк результата будут выглядеть так:

pr.planet_id	pr.product_type	pr.sell_price	pr_from.buy_price	pr_from.amount	profit
103	1	142	66	865	6080
183	1	136	66	865	5600
301	1	134	66	865	5440
186	1	133	66	865	5360
135	1	132	66	865	5280
133	1	129	66	865	5040
81	1	128	66	865	4960
83	1	128	66	865	4960
300	1	126	66	865	4800
82	1	125	66	865	4720

Получается, что максимальная прибыль равна 6080 галактических кредитов. Её можно получить, если купить 80 тонн минералов и продать их на планете Оннд.

Также можно было написать простые запросы вида **SELECT** * **FROM** [название таблицы] для всех указанных в условии таблиц и обработать данные вручную или воспользоваться библиотекой для доступа к SQL БД для используемого языка программирования, скачав представленный в условии файл с базой данных.

8. Технологии программирования (3 балла)

[Расписание станков]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	3 секунды
Ограничение по памяти	256 мегабайт

Известный завод ждет реорганизация — его собираются переоборудовать n новейшими станками. Перед тем, как эти станки будут установлены, требуется разработать интерфейс для обработки задач на этих станках.

После реорганизации наладчик завода будет распределять поступающие задачи между станками. У каждой задачи есть длительность. У каждого станка есть независимая очередь задач, причем новую задачу можно добавить либо строго в конец этой очереди, либо строго в начало, если это *срочная задача*.

Станки работают строго в порядке очереди, обрабатывая задачи подряд. Как только станок завершает задачу, он сразу же начинает работу над следующей в очереди, если такая есть. На переключение между задачами время не тратится.

Формально определим время начала работы над задачей как первый момент времени, после которого прогресс по задаче строго увеличится. Так, если в момент времени t на свободный станок приходят сначала обычная задача A и затем срочная задача U, временем начала U станет t, а временем начала A — момент завершения U.

Аналогично, время завершения работы над задачей — первый момент времени, когда прогресс по задаче достигает ее длительности. Так, если в момент времени t прогресс по задаче A достиг ее длительности, и станку поступил запрос на обработку срочной задачи U, временем завершения A все равно будет t.

Для большего понимания советуем после прочтения условия ознакомиться с иллюстрацией внизу.

Всего поддерживается четыре типа запросов.

1. Добавить задачу номер i длительностью d в конец очереди k-го станка. Если его очередь до этого была пустой, станок тут же начинает работу над добавленной задачей.

2. Отменить задачу с номером i. Если эта задача еще не была начата, она удаляется из очереди, а порядок остальных задач в очереди не меняется.
   Если эта задача уже была начата, работа над ней тут же останавливается, и станок берет в работу следующую задачу из очереди. Если эта задача уже была завершена или такой задачи не было, запрос отмены игнорируется
3. Добавить *срочную задачу* с номером i длительностью d в начало очереди k-го станка. В этом случае работа над текущей задачей (если она есть) на k -м станке приостанавливается с сохранением прогресса, и в работу берется данная срочная задача.
   До момента завершения срочной задачи все запросы добавления или отмены новых задач к станку номер k **игнорируются** для экономии ресурсов. Иными словами, если в момент времени t поступил запрос добавления срочной задачи, все следующие запросы, поступающие к k-му станку в моменты времени с t **включительно** до $t + d$ **не включительно** будут проигнорированы.
   По завершении срочной задачи, если работа над какой-то обычной задачей в очереди была приостановлена, эта задача без задержек возвращается в обработку.
4. Вывести ожидаемый момент времени завершения работы k-го станка.
   Для станка без задач в очереди «ожидаемым» временем завершения его работы будем считать текущий момент времени.

Еще раз **обратите внимание**, что во время исполнения срочной задачи станок игнорирует только все запросы добавления и отмены задач (включая добавление другой срочной задачи). Про запрос отмены будем считать, что он идет к тому станку, в очереди которого лежит соответствующая задача. Запросы вывода ожидаемого момента завершения работы **не игнорируются**.

Вам необходимо помочь наладчику и вывести для каждой задачи время начала и завершения работы над ней.

Формат входных данных

В первой строке дано целое число T $(1 \le T \le 1000)$ — количество наборов входных данных.

В первой строке описания набора входных данных через пробел даны два целых числа n и q $(1 \le n, q \le 10^5)$ — количество станков и количество запросов. Гарантируется, что сумма n и сумма q по всем наборам входных данных обе не превосходят 10^5.

В следующих q строках описаны запросы к заводу в одном из следующих форматов:

1. «t $schedule$ i (d) on k» — добавить задачу на k-й станок;
2. «t $cancel$ i» — отменить задачу;
3. «t $urgent$ i (d) on k» — добавить срочную задачу на k-м станке;
4. «t $expectation$ k» — узнать текущее ожидаемое время завершения работы k-го станка.

Параметр t — момент совершения запроса (целое неотрицательное число от 0 до 10^9). Для всех запросов выполняется $1 \le k \le n$, $1 \le i \le 10^9$ и $1 \le d \le 10^9$.

Также гарантируется, что номера задач (i) уникальны и не повторяются, а запросы упорядочены по времени, то есть t каждого следующего запроса не меньше, чем t предыдущего.

Формат выходных данных

Выведите в отдельной строке текущее ожидаемое время работы соответствующего станка после каждого запроса типа «$expectation$».

Затем выведите по строке на каждую задачу, запрос на добавление которой не был проигнорирован. Задачи должны быть упорядочены по возрастанию их номеров. В каждой строке через пробел должны быть выведены три числа: номер задачи, время начала ее обработки и время конца ее обработки (или отмены, если задача была отменена до завершения).

Для задач, которые были отменены до начала работы над ними, считайте время начала равным времени первого запроса их отмены.

Пример

Стандартный ввод	**Стандартный вывод**
3 2 2 0 schedule 1 (10) on 1 4 schedule 2 (3) on 2 3 6 0 schedule 1 (5) on 1 0 schedule 2 (5) on 2 2 urgent 3 (5) on 1 6 expectation 1 6 expectation 2 6 expectation 3 2 17 0 schedule 1 (10) on 1 0 schedule 2 (6) on 2 2 schedule 3 (5) on 1 3 schedule 4 (100) on 1	1 0 10 2 4 7 10 6 6 1 0 10 2 0 5 3 2 7 113 16 1 0 10 2 0 12 3 5 5 4 13 14 5 12 13 6 5 11

Стандартный ввод	Стандартный вывод
3 schedule 5 (1) on 2 5 cancel 3 5 urgent 6 (6) on 2 8 cancel 6 9 cancel 6 10 cancel 6 10 schedule 7 (150) on 2 10 urgent 8 (3) on 1 11 schedule 9 (3) on 2 14 expectation 1 14 cancel 4 14 schedule 10 (2) on 1 14 expectation 1	8 10 13 9 13 16 10 14 16

Замечание

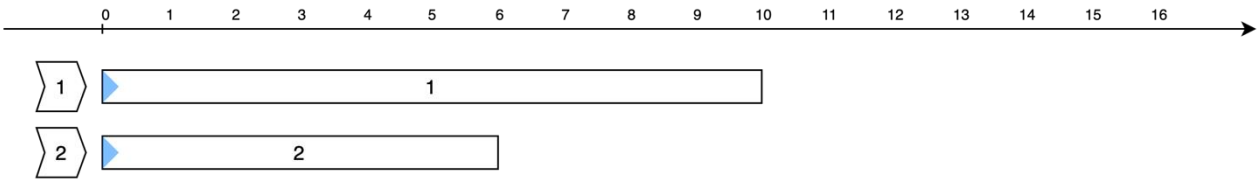
В первом примере из условия обе задачи будут обработаны по расписанию: с 0 до 10 и с 4 до 7 соответственно.

Во втором примере:

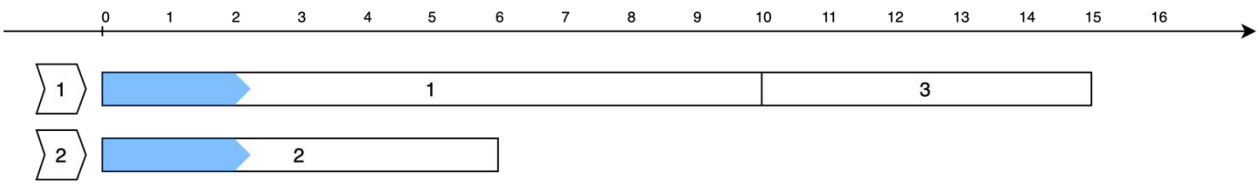
1. первая задача должна обрабатываться на первом станке в период $[0, 5]$;
2. вторая задача должна обрабатываться на втором станке в период $[0, 5]$;
3. в момент времени 2 срочная задача номер 3 добавляется на первый станок; она будет обрабатываться в период $[2, 7]$;
4. в момент времени 6 первому станку нужно еще 4 единицы времени на завершение задач 4 и 1; второй станок завершил работу в момент времени 5, а третий станок вообще не начинал работу — для них время ожидания до завершения работы равно 0.

Подробная иллюстрация к третьему примеру приведена ниже. Здесь синим обозначен прогресс по задачам, зеленым — завершенные задачи, красным — отмененные, а градиентом — срочные. Все интервалы короче 1 единицы времени (см. отмененную задачу 3) стоит воспринимать как интервалы длительностью 0.

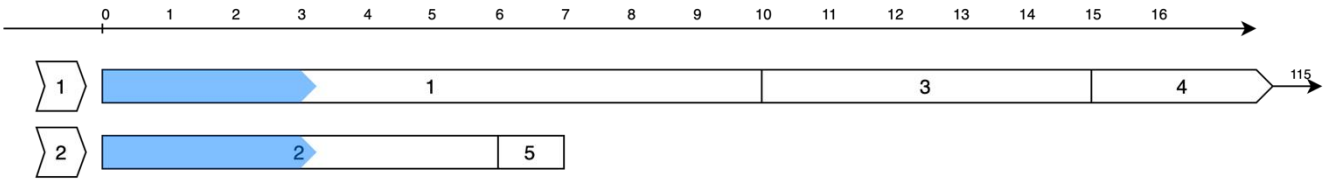
```
0 schedule 1 (10) on 1
0 schedule 2 (6) on 2
```



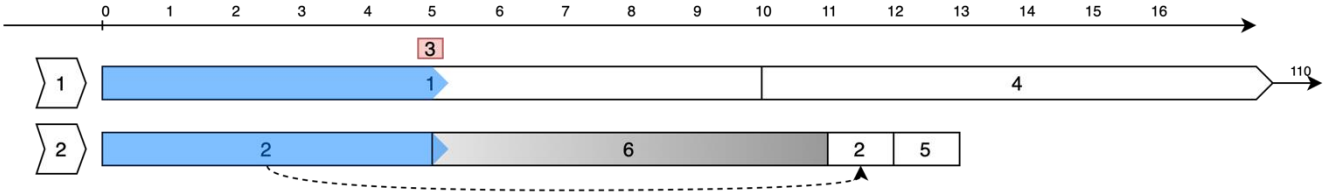

```
2 schedule 3 (5) on 1
```




```
3 schedule 4 (100) on 1
3 schedule 5 (1) on 2
```

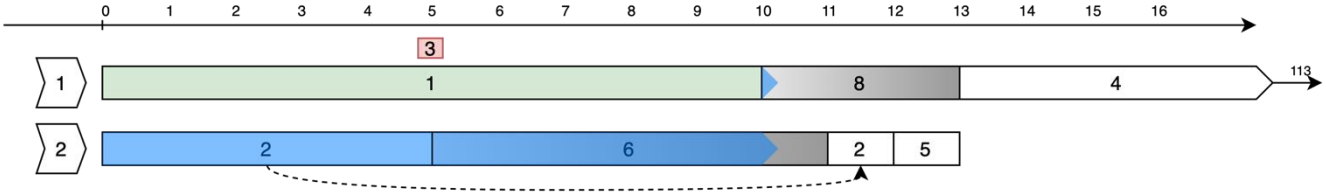



```
5 cancel 3
5 urgent 6 (6) on 2
```

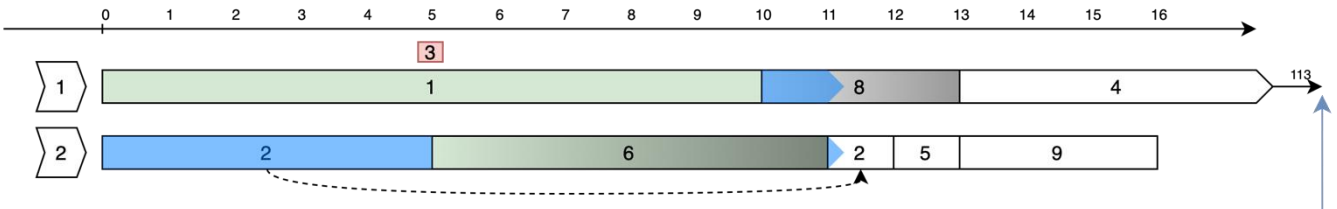



```
8 cancel 6
9 cancel 6
10 cancel 6
10 schedule 7 (150) on 2
10 urgent 8 (3) on 1
```

игнорируются, так как второй станок занят срочной задачей




```
11 schedule 9 (3) on 2
```

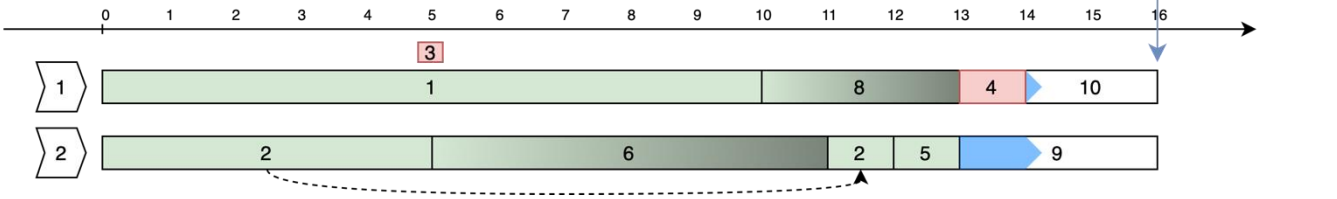



```
14 expectation 1
```

113

```
14 cancel 4
14 schedule 10 (2) on 1
14 expectation 1
```

16



Решение:

Это по большей части задача на реализацию, требующая при этом воспользоваться идеей ленивых вычислений. Разберем ее по частям.

Начнем с того, что, собственно, требуется хранить:

1. Для каждой задачи необходимо знать время начала и завершения ее обработки. Поскольку эти значения могут быть выставлены как последовательно при обработке, так и в любой момент времени при отмене, имеет смысл просто иметь доступ к задаче по ее номеру за $O(1)$: заведем словарь (unordered map, hash map) $jobs$, сопоставляющий номера задач и сами задачи.
2. Также для задачи полезно понимать, срочная ли она ($job.is\_urgent$), и какой прогресс по ней уже выполнен ($job.progress$). Ну и саму длительность задачи $job.duration$ тоже запомним отдельным полем. Определим $job.remaining()$ как 0, если она отменена, и как разницу между длительностью и прогрессом иначе.
3. Для каждого станка в любой момент времени может понадобиться узнать время его остановки. Мы будем хранить это отдельным полем $machine.finish$ для каждого станка, так как это самый эффективный способ быстро пересчитывать это значение при добавлении или отмене задач.
4. Разумеется, для каждого станка надо хранить указанную в условии очередь задач $machine.queue$.
5. И для обновления $machine.finish$ надо хранить словарь, сопоставляющий задачи станкам, на которых они обрабатываются.

Однако этого недостаточно. В этой задаче необходимы ленивые вычисления по одной причине: отмененная задача может находиться в середине очереди, из-за чего ее явное удаление оттуда может быть достаточно долгой операцией. Можно было бы в качестве альтернативы хранить упорядоченное множество вместо очереди, однако это замедлит и усложнит решение.

Идея ленивых вычислений в следующем: вместо реального удаления задач будем просто помечать задачу удаленной ($job.is\_cancelled$), а удалим ее из очереди, когда очередь до нее дойдет. Вторая идея — обрабатывать задачи не в режиме реального времени, а по необходимости. Действительно, станки независимы, поэтому можно проделать всю работу станка между предыдущим и текущим моментом времени только тогда, когда есть необходимость, например, понять где сейчас начало очереди. Поэтому будем дополнительно для каждого станка хранить последний момент времени, на котором мы «остановились» $machine.t$.

Тогда заведем следующие методы:

```
func (machine) work_until(time):
    while machine.t < time and len(machine.queue) > 0:
        job = machine.queue[0]
        if job.is_cancelled:
            machine.queue.popleft()
            continue

        machine.process_job(job, time)
        if job.remaining() == 0:
            machine.queue.popleft()

    machine.t = time
```

Иными словами, при необходимости в момент времени t узнать актуальное состояние станка, выполним всю его работу до этого момента: будем увеличивать прогресс по работам по очереди, пока не дойдем до текущего момента. При этом $process\_job$ выглядит следующим образом:

```
func (machine) process_job(job, time):
    rem = min (time - machine.t, job.remaining())
    if job.start is None:
        job.start = machine.t

    job.progress += rem
    machine.t += rem
    if job.remaining():
        job.finish = machine.t
```

Здесь мы увеличиваем прогресс по задаче на минимум из того, сколько осталось времени до ее завершения и до достижения момента $time$. Не забываем при этом выставить начало и конец обработки задачи в соответствии с определениями из условия.

Остается только аккуратно обработать все запросы. Перед выполнением запроса к станку вызываем для него $work\_until$, проверяем, что запрос не игнорируется, добавляем задачу в очередь и увеличиваем $machine.finish$ на ее длительность. При отмене задачи просто помечаем задачу отмененной, выставляем ей конец обработки и, возможно, начало, после чего уменьшаем $machine.finish$ для соответствующего станка на $job.remaining()$.

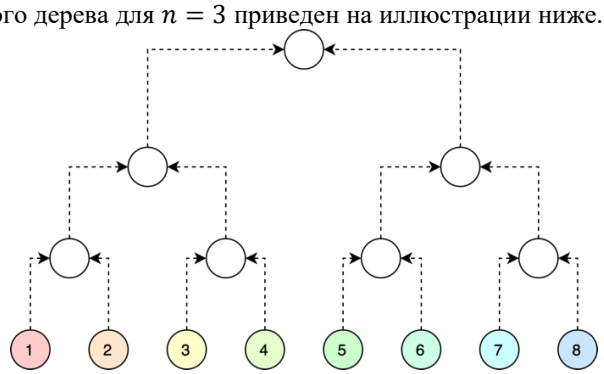
В итоге получаем решение, работающее за время $O(n + q)$, хотя в некоторых тестах и решение за $O(nq)$ могло пройти по времени.

9. Технологии программирования (4 балла)

[Бинарная Сила]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	4 секунды
Ограничение по памяти	256 мегабайт

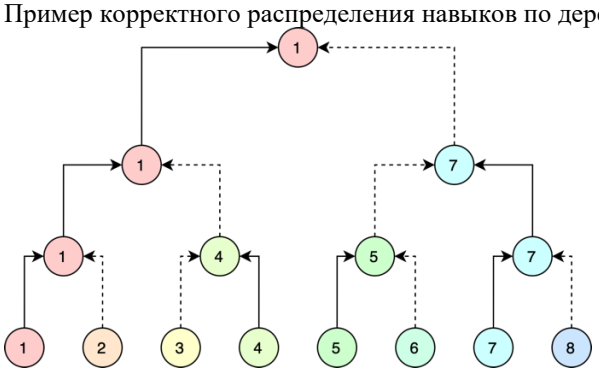
Вы играете в игру «Бинарная Сила» и управляете персонажем, у которого есть $d = 2^n$ навыков, пронумерованных 1 до d. Эти навыки расположены на листьях полного двоичного дерева высоты n, изначально все навыки имеют уровень 1. Пример такого дерева для $n = 3$ приведен на иллюстрации ниже.



После этого вы начинаете прокачивать навыки следующим образом.

- Навыки прокачиваются посредством заполнения двоичного дерева снизу вверх.
- Для очередной вершины дерева вы должны выбрать и записать в нее один из двух навыков, записанных в непосредственных детях этой вершины (на рисунке из детей в родителя ведут стрелки).
- *Уровнем* навыка считается число вершин, в которых выбран этот навык.

Пример корректного распределения навыков по дереву для $n = 3$ приведен ниже.



В этом примере первый навык имеет уровень 4, седьмой – уровень 3, четвертый и пятый – уровень 2, а второй, третий, шестой и восьмой не были прокачаны ни разу, поэтому остались на уровне 1.

Кроме прокачки персонажа, в игре есть m различных квестов, с помощью которых можно получать монетки. Квесты активируются после того, как все дерево навыков было заполнено.

Квесты бывают трех типов:

1. «$less\ x_i\ k_i\ s_i$» – вы получите s_i монет, если уровень навыка с номером x_i окажется **строго меньше** k_i.
2. «$exact\ x_i\ k_i\ s_i$» – вы получите s_i монет, если уровень навыка с номером x_i окажется **равен** k_i.
3. «$less\ x_i\ k_i\ s_i$» – вы получите s_i монет, если уровень навыка с номером x_i окажется **строго больше** k_i.

Так как монеты – очень ценный ресурс в игре «Бинарная Сила», вы хотите узнать максимальное количество монет, которое возможно получить с помощью имеющихся квестов после улучшения всех навыков.

Формат входных данных

Каждый тест состоит из нескольких независимых наборов входных данных. Первая строка содержит одно целое число t – количество наборов входных данных ($1 \le t \le 10^4$). Далее следует описание наборов входных данных.

Каждый набор начинается со строки, содержащей два целых числа n и m – высоту дерева навыков и количество квестов соответственно ($1 \le n \le 15$; $0 \le m \le 50\,000$). Число навыков при этом равно $d = 2^n$.

Далее следуют m строк, i-я из которых содержит четыре целых числа t_i, x_i, k_i, s_i – тип квеста и его описание ($1 \le t_i \le 3$; $1 \le x_i \le d$; $1 \le k_i \le n$; $1 \le s_i \le 10^9$). Типы квестов следуют в том же порядке, в котором они перечислены в условии: $t_i = 1$ соответствует квесту типа «$less$», $t_i = 2$ – квесту типа «$exact$» и $t_i = 3$ – квесту типа «$more$».

Гарантируется, что сумма d по всем наборам входных данных не превосходит 2^{16} и сумма m по всем наборам входных данных не превосходит $50\,000$

Формат выходных данных

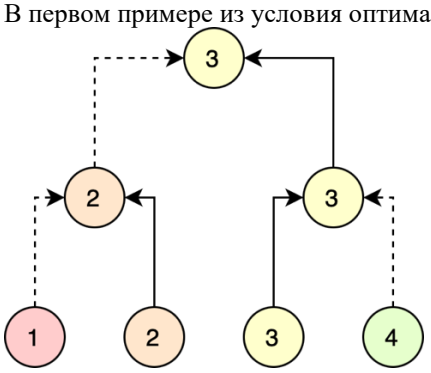
Для каждого набора выходных данных в отдельной строке выведите единственное число – максимальное количество монет, которые можно заработать.

Пример

Стандартный ввод	Стандартный вывод
4 2 5 2 1 1 10 2 2 2 100 1 2 2 5 3 3 1 15 2 3 1 10 1 3 2 2 1 5 2 1 1 10 3 1 1 3 10 6 2 1 1 1 2 2 2 1 2 4 3 1 2 8 4 1 2 16 5 1 2 32 6 1 10 4 1 6 3 5 1 6 4 7 2 6 5 11 3 6 1 3	125 10 6 15

Замечание

В первом примере из условия оптимальное распределение навыков выглядит следующим образом:



Действительно, мы получаем 100 монет за то, что второй навык имеет уровень **ровно** два, заодно получаем еще 10 монет за то, что первый имеет уровень **ровно** один. В этом случае третий квест не может быть выполнен, а из двух оставшихся мы получаем наибольшую награду (15 монет), если третий навык имеет уровень больше 1.

В третьем примере из условия достаточно для каждой вершины выбрать навык из правого ребенка. Тогда уровень навыка с номером 2^i будет в точности равен $i + 1$, и все квесты будут выполнены.

Решение:

В этой задаче можно было попробовать использовать жадные идеи — стараться сначала выполнять более дорогие квесты, но они не дали бы оптимальный ответ в большой части случаев. Так что будем оптимизировать полный перебор всех вариантов с помощью динамического программирования.

Динамическое программирование позволяет нам заметить, что если бы мы делали полный перебор, то при выборе навыка в очередной вершине нам были бы важны только значения навыков в ее детях, но не ниже. Соответственно, достаточно не перебирать все возможные распределения навыков по дереву, а только для каждой пары из вершины и ее возможного навыка определить оптимальный ответ на ее поддереве. Введем $dp[v][i]$ — оптимальный ответ на поддереве v, если в ней стоит навык i.

Теперь надо научиться пересчитывать эти значения. Порядок пересчета понятен — просто считаем снизу вверх по дереву, тогда при вычислении $dp[v]$ значения dp для ее детей уже посчитаны. Теперь заметим, что случаи, когда i находится в левом поддереве v, и в правом, симметричны, поэтому далее мы разберем только первый случай, а второй будет аналогичен.

Для удобства реализации пронумеруем все вершины сверху вниз и слева направо, начиная с нуля. Тогда несложно заметить, что дети вершины v будут иметь номера $2v + 1$ и $2v + 2$. При этом номер слоя вершины v будет равен $\lfloor log_2(v + 1) \rfloor$, что мы обозначим за $layer(v)$, а номера листьев в ее поддереве (в нумерации с нуля) будут лежать в интервале от $\min(v) = (v + 1) * 2^{n-layer(v)} - 2^n$ до $\max(v) = \min(v) + 2^{n-layer(v)}$ не включительно. Скажем, что $dp[v][i]$ отвечает за лист номер $\min(v) + i$.

При этом также сразу оценим, что сумма количества листьев по всем поддеревьям дерева имеет размер порядка $2^n * n$, так как каждый лист входит в n поддеревьев. Поэтому если мы будем пересчитывать $dp[v][i]$ за $O(1)$, полная асимптотика времени решения будет равна $O(2^n * n)$.

Пусть $i \in [min(v), \frac{min(v)+max(v)}{2})$, то есть находится среди левой половины листьев в поддереве v. Тогда $dp[v][i]$ подразумевает, что в вершине $2v + 1$ (в левом ребенке) также был выбран навык номер i. При этом в правом ребенке может быть выбран любой навык, поэтому

$$dp[v][i] = dp[2v + 1][i] + \max_{j \in [\min(2v+2), \max(2v+2))} dp[2v + 2][j] + REWARD$$

Здесь $REWARD$ означает количество монет, которое мы получим за размещение i-го навыка в v-й вершине. Давайте считать, что все квесты типа «$less$» изначально выполнены (то есть в базе динамики положим значения, равные сумме монет за все квесты типа «$less$» с уровнем больше 0), тогда за такое действие мы получим:

- минус число монет за квесты «$less$» для i-го навыка и уровня, соответствующего v (действительно, до этого ограничение квеста выполнялось, а теперь перестало);
- монеты за квесты «$exact$» для i-го навыка и уровня, соответствующего v;
- монеты за квесты «$more$» с тем же уровнем навыка по аналогичным причинам.

Чтобы быстро считать это число монет, заранее сгруппируем все квесты по навыкам, а в каждой группе — по уровню. И для каждого уровня сложим монеты из квестов «$exact$» и «$more$» и вычтем монеты за квесты «$less$». Это и будет значение $REWARD$, используемое выше.

Остается только заметить, что $\max_j dp[v][j]$ тоже можно поддерживать во время пересчета dp в отдельном массиве $\max\_dp$, добавив $O(1)$ времени на каждый пересчет. Тогда в сумме весь пересчет отработает за время $O(2^n * n)$, а ответ в конце будет находиться в $\max\_dp[0]$.

Заключительный этап, 9 класс (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (1 балл)

[WOW!]

Известно, что запись числа в шестнадцатеричной системе состоит из идущих подряд последовательностей четырех цифр KXYZ, то есть выглядит как KXYZKXYZ…KXYZ$_{16}$ (K, X, Y, Z – цифры 16-ричной системы счисления). После перевода этой записи в двоичную систему счисления оказалось, что она состоит из двоичных чисел W, разделённых двумя идущими подряд нолями, то есть выглядит как W00…00W00W$_2$, причем чисел W сколько же, сколько последовательностей KXYZ в исходном числе (W также может содержать в своей записи 00). Определите количество различных вариантов последовательностей цифр KXYZ, для которых это возможно. Два варианта последовательностей будут считаться различными, если отличаются значения хотя бы одной цифры, например варианты 1234, 1134 и 1324 – это три различных варианта. В ответе укажите целое число.

Ответ: 8192

Решение:

Вспомнив, что существует упрощённый способ перевода чисел из шестнадцатеричной системы счисления в двоичную, где каждой цифре шестнадцатеричного числа ставится в соответствие 4 цифры двоичной системы, после чего у первой цифры удаляются ведущие ноли, несложно сделать вывод, что каждый набор KXYZ, кроме первого, будет соответствовать ровно 16 знакам числа W00...00W00W. У результата перевода первого набора KXYZ, в отличие от остальных, будут удалены ведущие ноли. Так как каждое число W соответствует одному набору KXYZ, можно сделать вывод, что так как перед первым числом W нет ведущих нолей и оно соответствует первому набору KXYZ, то 00, разделяющие W являются ведущими нолями результата перевода KXYZ. Первые 2 цифры будут относиться к переводу K, то есть K должно в двоичной системе иметь вид 0010 или 0011, т.е. иметь ровно 2 ведущих ноля. Для остальных цифр у нас ограничения не накладываются, то есть они могут принять любое из 16 возможных значений. Итого получим 2 * 16 * 16 * 16 = 8192.

2. Кодирование информации. Объем информации (3 балла)

[Алиса и Боб]

Алиса решила написать генератор простых текстов на английском языке. Для хранения текста в памяти она решила использовать неравномерное кодирование, основываясь на частоте букв в английском алфавите. Она воспользовалась простой таблицей:

Буква	**E**	**T**	**A**	**O**	**I**	**N**	**S**	**H**	**R**	**D**	**L**	**C**	**U**
Частотность, %	12,7	9,06	8,17	7,51	6,97	6,75	6,33	6,09	5,99	4,25	4,03	2,78	2,76
Буква	**M**	**W**	**F**	**G**	**Y**	**P**	**B**	**V**	**K**	**X**	**J**	**Q**	**Z**
Частотность, %	2,41	2,36	2,23	2,02	1,97	1,93	1,49	0,98	0,77	0,15	0,15	0,1	0,05

Кодирование Алисы построено следующим образом:

1. В начале каждого кода символа находится специальный бит, обозначающий начало кода.
2. Затем идёт бит, отвечающий за то, является ли буква заглавной.
3. Кодом длиной i бит она кодирует 2^i букв с наибольшей популярностью, которым ещё не назначены более короткие коды. Например, буквы E и T получат однобитные коды, следующие 4 буквы – двухбитные коды, и так далее.

Её друг Боб справедливо заметил, что так как она генерирует простые тексты, в них используется не более 1500 различных слов (Боб также игнорирует регистр букв в слове) и предложил использовать минимальное, одинаковое для всех слов количество бит, а также для каждого слова записывать 1 бит метаинформации – начинается ли слово с заглавной буквы.

Определите, при какой минимальной средней длине слова способ Боба будет эффективнее способа Алисы (для хранения всего текста потребуется строго меньше бит). Средней длиной слова будем считать количество символов, поделенное на количество слов и округлённое до ближайшего целого числа. В ответ запишите одно число – искомую среднюю длину.

Ответ: 3

Решение:

Определим, какое количество бит будет соответствовать каждой из букв. Буквы E и T получат коды длиной 1 бит. Буквы A, O, I, N - длиной 2 бита, буквы S-M получат коды длиной 3 бита, оставшиеся буквы - длиной 4 бит.
Возьмём общее количество букв в тексте за N. Тогда каждая буква встретится $N*x_i/100$ раз, где x_i - частотность буквы i-ой буквы. Общий вес текста будет вычисляться как $N \cdot \sum_{i=0}^{25}(2 + c_i) \cdot \frac{x_i}{100}$, где c_i - длина кода i-ой буквы. Подставив все необходимые значения получим $4{,}4128 \cdot N$.

Вычислим ответ для способа Боба: в его случае ответ будет вычисляться как $\left((\log_2 1500) + 1\right) \cdot \frac{N}{A}$, где N - число букв в тексте, A - искомая средняя длина, $(\log_2 1500)$ - округленный вверх логарифм. Упросим выражение и получим $\frac{12 \cdot N}{A}$.

Составим итоговое неравенство: $4{,}4128 \cdot N > \frac{12 \cdot N}{A}$. Сократив данное выражение, получим, что $A > \frac{12}{4{,}4128}$, где правая часть выражения примерно равна 2,716, таким образом минимальное A, большее этой величины будет равно 3.

3. Основы логики (2 балла) [Под знаком вопроса]

Дано логическое уравнение с тремя логическими переменными A, B и C:

$$(A\ ?\ B)\ ?\ (B\ ?\ C)\ ?\ (A\ ?\ C) = 0.$$

В данном уравнении на месте символа ? может стоять знак операции конъюнкции или дизъюнкции. Сколько существует способов заменить ? так, чтобы уравнение имело ровно 4 решения?

Ответ: 2

Решение:

Для начала заметим, что всего возможных решений 2^5, т.е. 32 варианта. Чтобы найти ответ на данную задачу необходимо найти все варианты, при которых решений ровно 4 и доказать, что больше решений быть не может. Однако, очевидно, что хотелось бы избежать построения 32 таблиц истинности. Также выпишем все возможные варианты и будет вычёркивать их по мере рассмотрения (+ будет обозначать дизъюнкцию, * - конъюнкцию.

Для удобства рассуждений также пронумеруем знаки вопроса слева направо числами от 1 до 5.

12345	12345	12345	12345
+++++	+*+++	*++++	**+++
++++*	+*++*	*+++*	**++*
+++*+	+*+*+	*++*+	**+*+
+++**	+*+**	*++**	**+**
++*++	+**++	*+*++	***++
++*+*	+**+*	*+*+*	***+*
++**+	+***+	*+**+	****+
++***	+****	*+***	*****

Обратим внимание, что если в одной из крайних скобок (на позициях 1 или 5) будет стоять дизъюнкция и на соседней с ней позиции (т.е. 2 или 4 соответственно) будет также стоять дизъюнкция, то итоговое выражение будет иметь минимум 6 единиц вне зависимости от остальных знаков, т.е. максимум два ноля. Выражений, имеющих на позициях 1 и 2 дизъюнкции 2^3, т.е. 8. Также вычислим, что выражений, имеющих в начале две конъюнкции или конъюнкцию и дизъюнкцию (в любом порядке) и две дизъюнкции в конце будет (2 * 2 – 1) * 2 * 1 * 1 = 6 вариантов. Итого 14 вариантов, которые нам точно не подходят. Отметим красным отсечённые варианты

12345	12345	12345	12345
+++++	+*+++	*++++	**+++
++++*	+*++*	*+++*	**++*
+++*+	+*+*+	*++*+	**+*+
+++**	+*+**	*++**	**+**
++*++	+**++	*+*++	***++
++*+*	+**+*	*+*+*	***+*
++**+	+***+	*+**+	****+
++***	+****	*+***	*****

Также отсечём вариант, где дизъюнкции стоят на 2, 3 и 4 позициях (а на остальных - конъюнкции, т.к. варианты, где слева или справа две дизъюнкции подряд уже учтены) - очевидно, что в таком случае также вне зависимости от результатов в

крайних скобках, результат в центральной и то, что с ним проводится только дизъюнкция, не позволит нам получить более 2 нолей.

Заметим, что если в выражении будет всего одна дизъюнкция, то она будет на позиции 1, 3 или 5 позиции, то выражение будет обращаться в ноль во всех случаях, кроме A=B=C=1. А если она будет на позиции 2 или 4, то единица будет достижима, если единице будет равна соответствующая крайняя скобка, т.е. либо дизъюнкция на позиции 2, A=B=1 и C – любое, либо дизъюнкция на позиции 4 и A=C=1, B – любое. То есть, варианты, где всего одна дизъюнкция, которых ровно 5, нам не походят. Также отметим, что вариант, где дизъюнкций нет вовсе нам тоже не подойдёт - там также ноли во всех случаях, кроме A=B=C=1.

12345	12345	12345	12345
+++++	+*+++	*++++	**+++
++++*	+*++*	*+++*	**++*
+++*+	+*+*+	*++*+	**+*+
+++**	+*+**	*++**	**+**
++*++	+**++	*+*++	***++
++*+*	+**+*	*+*+*	***+*
++**+	+***+	*+**+	****+
++***	+****	*+***	*****

На текущий момент остаётся 32-14-1-5-1=11 вариантов. Заметим, что среди них нет вариантов более чем с 3-мя дизъюнкциями, т.к. все такие варианты были отсечены как имеющие две дизъюнкции в начале или в конце.

Вариантов с ровно тремя дизъюнкциями осталось ровно 3 - на позициях 1, 3, 4, или 1, 3, 5 или 2, 3, 5. Построив таблицы истинности для этих трёх выражений выясним, что варианты 1, 3, 4 и 2, 3, 5 дают по 3 ноля, а вот вариант 1, 3, 5 даёт ровно искомые 4 ответа.

12345	12345	12345	12345
+++++	+*+++	*++++	**+++
++++*	+*++*	*+++*	**++*
+++*+	+*+*+	*++*+	**+*+
+++**	+*+**	*++**	**+**
++*++	+**++	*+*++	***++
++*+*	+**+*	*+*+*	***+*
++**+	+***+	*+**+	****+
++***	+****	*+***	*****

Во всех оставшихся вариантах по 3 конъюнкции. Заметим, что по принципу Дирихле хотя бы одна из них окажется внутри скобок. Рассмотрим вариант, где только одна из конъюнкций внутри скобок. Тогда другие две - между скобок, на позициях 2 и 4. Тогда мы получим 0 во всех случаях, когда он получается в скобках с конъюнкцией, то есть в 6 случаях из 8 вне зависимости от того, в какую скобку мы ставим конъюнкцию. То есть, мы отсекаем ещё 3 варианта. Рассмотрим вариант, где внутри скобок окажутся все 3 конъюнкции. То есть, они будут на позициях 1, 3 и 5. В этом варианте в таблице истинности будет ровно 4 ноля.

12345	12345	12345	12345
+++++	+*+++	*++++	**+++
++++*	+*++*	*+++*	**++*
+++*+	+*+*+	*++*+	**+*+
+++**	+*+**	*++**	**+**
++*++	+**++	*+*++	***++
++*+*	+**+*	*+*+*	***+*
++**+	+***+	*+**+	****+
++***	+****	*+***	*****

Остаются 4 варианта – заметим, что во всех них мы проводим дизъюнкцию результата конъюнкции двух переменных и остального выражения, которое в свою очередь состоит из конъюнкции результатов конъюнкции и дизъюнкции. То есть, все эти варианты будут иметь одинаковое число нолей в таблице истинности. Посчитав для любого из них таблицу истинности выясним, что нолей будет 5, т.е. найденный ответ не увеличится. Итого ответ 2.

4. Алгоритмизация и программирование. Формальные исполнители (1 балл)

[Простые замены]

Дана последовательность нолей и единиц вида 00110011..., длиной 5000 символов. За один ход можно либо заменить 3 подряд идущих символа, начиная с текущего, на противоположные (0 на 1, 1 на 0), либо перейти к следующему символу. В случае если от некоторого символа до конца строки меньше 3 символов, замена производится для всех оставшихся до конца строки символов (например, строку 100 можно превратить в строку 111, сделав переход к следующему символу, и затем заменив 00 на 11). Для первого хода, текущим считается первый символ строки. Сколько раз понадобится совершить ход с заменой, чтобы превратить эту строку в состоящую только из единиц?

Ответ: 2502

Решение:

Чтобы понять ход работы алгоритма, применим его на меньшей строке - например, на строке из 20 символов. Для простоты будем выписывать результаты только искомых шагов с заменой, выделяя заменяемую подстроку жирным:

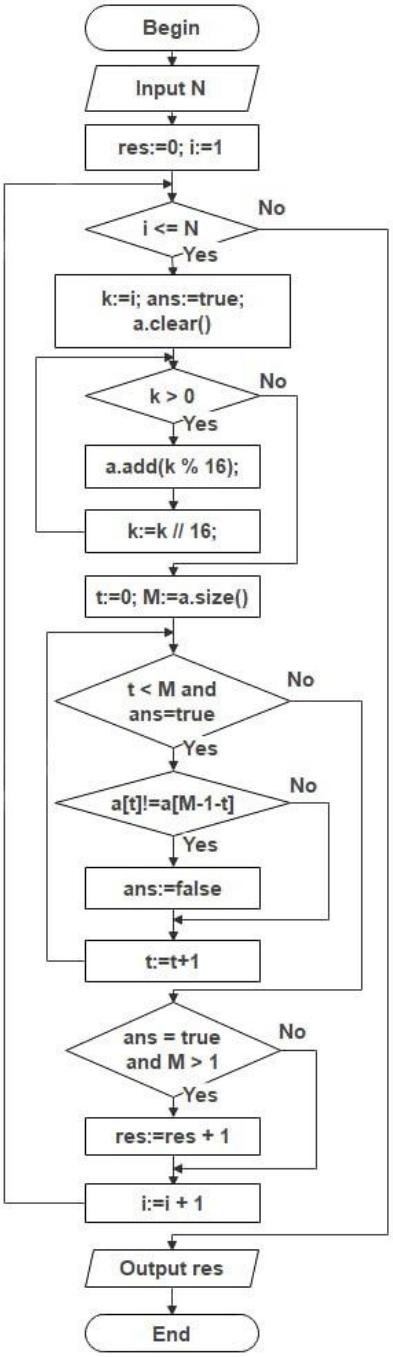
1. **001**10011001100110011
2. 11**010**011001100110011
3. 111**010**11001100110011
4. 1111**011**1001100110011
5. 11111**001**001100110011
6. 1111111**000**1100110011
7. 11111111111100110011

Заметим интересную закономерность: первые 12 символов превращаются в строку из единиц ровно за 6 шагов. Также заметим, что последующие 12 символов в точности равны первым 12 символам, а значит тоже потребуется ровно 6 шагов, и так далее. Значит, первые 5000//12 чисел буду преобразованы за (5000//12)*6=2496 шагов. Останется ещё 8 символов, то есть строка 00110011. Воспользовавшись вычислениями выше, заметим, что для 8 символов также нужно 6 шагов, т.е. всего нужно 2496+6=2502 шага.

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (3 балла)

[Поиск мина и макса]

Дана блок-схема алгоритма:



Операция **%** обозначает взятие остатка от деления, операция // обозначает целочисленное деление. Операция **a.clear()** создаёт пустой список чисел, операция **a.size()** возвращает количество чисел в **a**, операция **a.add(x)** дописывает **x** в конец списка **a**. На вход было подано некоторое натуральное число **N**, после чего алгоритм вывел 278. Найдите минимальное и максимальное **N**, при которых это возможно. В ответ запишите через пробел два числа – минимальное и максимальное значения.

Ответ: 9826 10097

Решение:

Для решения задачи реализуем описанный алгоритм. Нетрудно понять, что он вычисляет количество чисел, меньших либо равных N, которые при переводе в шестнадцатеричную систему счисления будут иметь вид палиндрома длиной 2 или больше. Очевидно, что минимальное число, при котором можно получить ответ 278 — это 278-й палиндром длины 2 или более, а максимальное – это предыдущее число перед 279-м палиндромом. Нетрудно вычислить, что двузначных шестнадцатеричных палиндромов - 15, трёхзначных - 15 * 16 = 240, значит искомый палиндром - 23-й четырёхзначный. Четырёхзначных палиндромов, начинающихся с 1 – 16, значит, нам нужен 7-й, начинающийся с 2, т.е. 2662. $2662_{16}=9826_{10}$. Следующий палиндром – это 2772, т.е. искомое максимальное $N=2771_{16}=10097_{10}$. Итого ответ - 9826 10097.

6. Телекоммуникационные технологии (2 балла)
[Последствия невнимательности]

Маска сети для IPv4 адресации – это 4-байтное число, которое делит IP адрес на адрес сети (первая часть) и адрес узла (вторая часть). У всех адресов одной IP-сети совпадают первые части и отличаются вторые. Для части IP адреса, соответствующей адресу сети, в маске сети содержатся двоичные единицы, а для части IP адреса, соответствующей адресу узла, в маске сети содержатся двоичные нули.

Недавно Олег узнал о масках подсети, однако, пока ему рассказывали, он был не слишком внимателен, и запомнил, что чтобы получить адрес сети, нужно применить побитовую операцию «исключающее или» к адресу узла и маске. В результате из адреса узла X и трёх различных корректных масок A, B и C Олег получил следующие “адреса сетей” – 40.102.219.169, 40.102.196.89 и 40.102.59.169. Определите адрес узла, который использовал Олег. В ответ запишите 4 числа без ведущих нолей, разделённых точками – адрес узла.

Пример записи ответа: 192.168.0.1

Ответ: 215.153.59.169

Решение:

Для решения данной задачи переведём все 3 “адреса сети” в двоичный формат:

	Первый байт	Второй байт	Третий байт	Четвёртый байт
Адрес от маски A	00101000	01100110	11011011	10101001
Адрес от маски B	00101000	01100110	11000100	01011001
Адрес от маски C	00101000	01100110	00111011	10101001
Восстановленный адрес				

Обратим внимание, что у всех адресов совпадают начала – значит, для всех совпадающих бит начала был применён XOR с одним и тем же значением, так как это начало маски - с единицей. Применив к этим битам XOR с 1, повторно получим начальный фрагмент исходного адреса

	Первый байт	Второй байт	Третий байт	Четвёртый байт
Адрес от маски A	00101000	01100110	11011011	10101001
Адрес от маски B	00101000	01100110	11000100	01011001
Адрес от маски C	00101000	01100110	00111011	10101001
Восстановленный адрес	11010111	10011001		

Повторим аналогичную операцию с совпадающими концами, но с одним отличием так как это конец маски, то нужно применять XOR с 0.

	Первый байт	Второй байт	Третий байт	Четвёртый байт
Адрес от маски A	00101000	01100110	11011011	10101001
Адрес от маски B	00101000	01100110	11000100	01011001
Адрес от маски C	00101000	01100110	00111011	10101001
Восстановленный адрес	11010111	10011001		0110

Обратим внимание на третий байт: так как первые 3 бита совпадают у масок A и B, и при этом все 3 маски различны и все маски состоят сначала из последовательности единиц, а затем из последовательности нолей, очевидно, что нули не могли начаться одновременно у масок A и B (это противоречит условию о неравенстве масок), следовательно нули начались у маски C. Используя это знание, можно из 3-го адреса полностью восстановить искомый адрес: 11010111.10011001.00111011.10101001, т.е. 215.153.59.169.

7. Технологии обработки информации в электронных таблицах, технологии сортировки и фильтрации данных (2 балла)
[Есть тут кто?]

В ячейки таблицы A1, A4, B1, B2, C1, D1 поместили некоторые формулы (см. изображение).

	A	B	C	D	E
1	=COUNTIF(B2:AD30; "=0")	=A2	=A3	=IF(AND((B$1>C$1); (C$1>1)); C$1-1; IF(C$1<10; C$1+1; C$1-1))	
2	[redacted]	=MOD(MOD((B1+A2); 100); 11)			
3					
4	=IF(AND(($A2>$A3); ($A3>1)); $A3-1; IF($A3<10; $A3+1; $A3-1))				
5					

Формулу из ячейки B2 скопировали во все ячейки диапазона B2:AD30. Формулу из ячейки A4 скопировали во все ячейки диапазона A4:A30. Формулу из ячейки D1 скопировали во все ячейки диапазона D1:AD1.

В каждую из ячеек A2 и A3 либо поместили число от 1 до 10, либо оставили ячейку пустой. В результате в ячейке A1 отобразилось значение 206, число в ячейке B2 оказалось нечётным, число в ячейке B3 совпало с числом в ячейке B2. Определите для каждой из ячеек A2 и A3, поместили ли туда число (и тогда какое) или оставили её пустой. Если ячейка

осталась пустой, напишите -, в противном случае укажите число. В ответ запишите через пробел два значения – значение в ячейке A2 и значение в ячейке A3.

Пример записи ответа: 12 -

Примечание: таблица соответствия имён используемых функций.

	Google Sheets	**Excel**	**LibreOffice**
Условное вычисление	IF	ЕСЛИ	IF
Логическое И	AND	И	AND
Остаток от деления	MOD	ОСТАТ	MOD
Число ячеек, для которых верно условие	COUNTIF	СЧЁТЕСЛИ	COUNTIFS

Ответ: 8 -

Решение:

В первую очередь воссоздадим указанную таблицу. Очевидно, что число в ячейке B2 - это сумма чисел в ячейках A2 и B1, взятая по модулю 100, а затем по модулю 11. Сразу заметим, то так как мы суммируем числа, не превосходящие 10 (11, если в суммировании участвуют ячейки из диапазона B2:AD30), взятие по модулю 100 не имеет никакого смысла. Так как число в ячейке B1 равно числу в ячейке A2, то изначальная сумма будет равна удвоенному A2. Следовательно, удвоенное A2 меньше 11 - оно и останется чётным, иначе чётность изменится. Следовательно, из того, что в B2 нечётное число, следует, что в A2 число от 6 до 10 включительно. Число в ячейке B3 является суммой чисел в ячейках B2 и A3, взятой по модулю 100, а затем по модулю 11. Так как число в B3 равно числу в B2, то число в B3, то число в ячейке A3 должно быть равно 0 по модулю 11. Ни одно из чисел от 1 до 10 не подходит под это определение, значит, в ячейке A3 числа нет, т.е. ответ для этой ячейки -. Установив значение (точнее, его отсутствие) в A3, значение в A2 можно восстановить простым перебором возможных значений. Только при установлении туда значения 8, число в A1 совпадёт с искомым 206. Итого ответ - 8 -.

8. Технологии программирования (3 балла)

[Покраска холста]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	1 секунды
Ограничение по памяти	256 мегабайт

Радиоуправляемый робот умеет перемещаться по клетчатому полю размером $n \times m$ (n строк и m столбцов) и красить клетки в один из четырех цветов (красный, зеленый, синий и белый). Будем обозначать клетку на пересечении i-й сверху строки и j-го слева столбца как (i, j).

Робот выполняет *команды* пользователя, при этом перемещаясь по полю в соответствии с заданными *настройками* и *ограничениями*.

Настройки представляют собой матрицу S размера $n \times m$, каждый элемент которой – либо $\emptyset$, либо пара из направления (вправо, вверх, влево, вниз) и цвета. Если $S_{i,j} = (d, c)$, то после того, как робот красит клетку (i, j) в какой-либо цвет, он сразу же смещается на одну клетку в направлении d и красит ее в цвет c. Если для новой покрашенной клетки $S_{i',j'} \neq \emptyset$, процесс продолжается по тем же правилам.

Ограничения бывают двух типов:

1. ограничение на максимальное разрешенное число клеток цвета c;
2. запрет наличия на поле квадрата 2×2, покрашенного цветами $\begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$.

Пользователю доступны следующие *команды* для взаимодействия с роботом:

1. «$color\ i\ j\ c$» – закрасить клетку (i, j) в цвет c, после чего выполнять действия в соответствии с настройками; процесс останавливается, когда
   - очередное перемещение привело робота за границу поля;
   - очередное перемещение привело робота в клетку, которую он уже красил в процессе выполнения текущей команды;
   - для очередной клетки $S_{i',j'} \neq \emptyset$;
   - с очередной покраской перестанет выполняться какое-то из ограничений.

Обратите внимание, что если первая же покраска клетки (i, j) в цвет c приводит к нарушению какого-то из ограничений, робот остановится сразу же, не покрасив ни одну клетку.

2. «$limit\ c\ x$» – выставить ограничение на максимальное число клеток цвета c в x. Если в настоящий момент на поле уже больше x клеток цвета c, команда игнорируется и ограничение не меняется. Для каждого цвета в каждый момент времени действует только последнее введенное на него ограничение на число клеток.

3. «$block\ c_{1,1}\ c_{1,2}\ c_{2,1}\ c_{2,2}$» – запретить появление квадратов 2×2, раскрашенных цветами $\begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$. Если в настоящий момент на поле уже есть квадрат, раскрашенный таким образом, команда игнорируется и ограничение не добавляется.

4. «$allow\ c_{1,1}\ c_{1,2}\ c_{2,1}\ c_{2,2}$» – аналогично, отменить запрет на раскрашенные соответствующим образом квадраты 2×2, если такой запрет сейчас есть.

5. «$settings\ i\ j\ d\ c$» – выставить настройки для клетки (i, j) в значение (d, c), где d – направление перемещения, а c – цвет, в который затем будет раскрашена клетка, в которую робот переместится.

6. «$settings\ i\ j\ none$» – выставить настройки для клетки (i, j) в значение $\emptyset$, соответствующее отсутствию перемещения после покраски клетки (i, j).

Еще раз повторим, что робот никогда не красит одну и ту же клетку дважды во время исполнения одной команды, а также останавливается до момента первого нарушения какого-либо ограничения. Например, если $S_{1,1} = (\rightarrow, red)$, $S_{1,2} = (\downarrow, blue)$, $S_{2,2} = (\leftarrow, green)$ и $S_{2,1} = (\uparrow, red)$, то при поступлении команды «$color\ 1\ 1\ blue$», робот покрасит $(1,1)$ в синий, $(1,2)$ в красный, $(2, 2)$ в синий и $(2, 1)$ в зеленый. Затем робот остановится, так как клетка $(1, 1)$ уже была покрашена при исполнении этой команды.

Изначально все настройки равны $\emptyset$, никакие ограничения не введены, а все клетки поля покрашены в белый цвет. Вам дан список из q команд пользователя, которые были последовательно отправлены роботу. Выведите раскраску поля после применения всех этих команд.

Формат входных данных

В первой строке записано целое число t $(1 \le t \le 1000)$ – число наборов входных данных в тесте.

В первой строке каждого набора данных даны три целых положительных числа n, m и q – размеры поля и число команд. Гарантируется, что сумма $n \cdot m \cdot q$ по всем наборам входных данных не превосходит 10^5.

В следующих q строках дано описание команд, посланных роботу в формате, описанном в условии. Направления задаются строками «$right$» (вправо), «up» (вверх), «$left$» (влево), «$down$» (вниз). Цвета задаются заглавными буквами 'R' (красный), 'G' (зеленый), 'B' (синий) и 'W' (белый).

Формат выходных данных

Для каждого набора входных данных выведите итоговую раскраску поля, полученную после обработки всех команд.

Пример

Стандартный ввод	**Стандартный вывод**
2 3 2 11 settings 1 1 down R settings 2 1 down R color 1 1 R limit B 0 settings 2 1 right B color 2 1 G block R W G R color 2 2 R settings 2 2 up G limit B 1 color 2 2 B 3 4 10 limit R 1 settings 2 1 right R settings 2 2 right B settings 2 3 down B settings 3 3 left B settings 3 2 up B color 2 1 G limit R 3 settings 2 4 up R color 2 4 R	R G G B R W W W W R G R B R W B B W

Замечание

В первом примере из условия

1. Команда «$color\ 1\ 1\ R$» красит $(1,1)$ в красный, после чего из-за $S_{1,1} = (\downarrow, red)$ клетка $(2,1)$ тоже красится в красный, а из-за $S_{2,1} = (\downarrow, red)$ затем и $(3,1)$ красится в красный.

2. При выполнении «$color\ 2\ 1\ G$» робот красит $(2,1)$ в зеленый. $S_{2,1}$ в этот момент уже равно $(\rightarrow, blue)$, поэтому после этого клетка $(2,2)$ должна быть покрашена в синий, но это бы нарушило ограничение «$limit\ B\ 0$», поэтому процесс останавливается до этого.

3. После этого поле выглядит как

RW
GW
RW

4. Затем «$color\ 2\ 2\ R$» должен покрасить $(2,2)$ в красный, но это бы привело к получению квадрата с цветами $RWGR$, который запрещен, поэтому выполнение команды сразу останавливается.

5. Последняя команда покраски «$color\ 2\ 2\ B$» отрабатывает корректно, так как до этого было разрешено иметь на поле не больше 1 синей клетки. После чего, в соответствии с $S_{2,2} = (\uparrow, green)$, клетка $(1,2)$ красится в зеленый.

6. Итоговый рисунок:

RG
GB
RW

Решение:

Эта задача на реализацию не требовала никаких алгоритмических идей. В ней было необходимо просто аккуратно реализовать все требования из условия, при этом ограничения по времени позволяли выполнять каждый запрос за время $O(nm)$ с итоговой асимптотикой времени решения $O(nmq)$.

Для начала поймем, какие данные нам надо хранить.

1. Во-первых, надо хранить настройки. Для этого достаточно было завести матрицу $settings$ размером $n \times m$, каждый элемент которой — либо какое-то особое значение, соответствующее $\emptyset$, либо реальная пара из направления и цвета (или новой клетки и цвета во втором варианте).

2. Во-вторых, ограничения. Ограничения первого типа можно хранить в словаре (unordered map, hash map) $limits$ из цветов в последний введенный лимит. Ограничения второго типа достаточно хранить просто в множестве из четверток цветов $block$.

3. Чтобы быстро понимать, не нарушаем ли мы ограничение на число клеток какого-либо цвета, будем поддерживать число клеток каждого цвета на поле в словаре $count$ из цветов в количество.

4. Ну и, разумеется, надо хранить само поле $field$.

И на самом деле, этого уже достаточно для полного решения. Опишем ниже процесс ответов на запросы (команды).

1. Запрос покраски. Перед обработкой заведем изначально пустое множество уже посещенных клеток. Затем в цикле $while$ будем совершать покраску и перемещения. Покраску проще всего расписать следующим псевдокодом:

$func\ color(i, j, c)$:
    $if\ count[c] + 1 > limit[c]$:
        $STOP$
    $if\ block.contains(any\ of\ \{$
        $[field[i-1][j-1], field[i-1][j], field[i][j-1], c],$
        $[field[i-1][j], field[i-1][j+1], c, field[i][j+1]],$
        $[field[i][j-1], c, field[i+1][j-1], field[i+1][j]],$
        $[c, field[i][j+1], field[i+1][j], field[i+1][j+1]],$
    $\}):$
        $STOP$
    $count[field[i][j]] - -$
    $field[i][j] = c$
    $count[c] + +$

То есть проверим, не будет ли нарушено никакое ограничение, после чего выполним покраску и обновим количество клеток двух затронутых цветов на поле. Затем уже выполним перемещение и проверку, что мы не вышли за границы поля, и не пришли в клетку, которая уже есть в нашем множестве посещенных.

2. Запросы добавления или удаления ограничения на квадраты транслируются просто в добавление или удаление четверки цветов из $block$. Единственное отличие – перед удалением ограничения ничего проверять не надо, а перед добавлением надо отдельно пройти по всему полю и для каждого квадрата проверить, не совпадает ли он с добавляемым ограничением. Это просто два вложенных цикла по строкам и столбцам поля.

3. Аналогично, запрос изменения лимита по цвету состоит из проверки текущего $count$ этого цвета и выставления нового значения в $limit$.

4. Ну и запросы изменения настроек просто меняют матрицу настроек без каких-либо дополнительных проверок. Направления (первый вариант) можно было хранить просто как пару из смещения по строкам и смещения по столбцам, что является стандартной практикой.

Таким образом,мы получаем полное решение, работающее за время $O(nmq)$.

9. Технологии программирования (4 балла)

[Есть тут кто?]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

На производстве расположены n роботов, пронумерованных от 1 до n. Любая пара роботов может быть либо соединена проводом, либо нет. Всего на производстве m проводов и i-й из них сейчас соединяет роботов с номерами a_i и b_i.

Вы находитесь рядом с роботом номер 1. Роботом номер t можно *управлять*, если он связан с первым некоторой цепочкой проводов, то есть

- либо если $t = 1$;
- либо если существуют такие $i_1, \dots, i_k$, что $i_1 = 1$, $i_k = t$, и любые два соседних в этой последовательности робота i_j и i_{j+1} связаны проводом.

Любому из роботов, которыми можно управлять, можно послать команду «извлеки второй конец подключенного к себе провода и подключи его к другому роботу». Иными словами, если роботом номер t можно управлять, и есть провод,

соединяющий его с роботом номер x, то можно заменить провод (t, x) на провод (t, y) для любого $y \neq t$, еще не связанного проводом с t. Обратите – внимание, что после этого вы можете потерять управление над t-м роботом, если единственная связь t-го с первым проходила через x-й.

Ваша задача – получить управление над максимально возможным количеством роботов на производстве. Найдите это количество и самую короткую последовательность действий, приводящую к такому исходу.

Формат входных данных

В первой строке записано целое число T $(1 \leq T \leq 1000)$ – количество наборов входных данных в тесте.

Первая строка каждого набора входных данных содержит целые числа n и m $(1 \leq n \leq 10^5;\ 0 \leq m \leq 1.5 \cdot 10^5)$ – количество роботов и количество проводов, соответственно.

В следующих m строках дано описание проводов: в i-й строке даны целые числа a_i и b_i $(1 \leq a_i, b_i \leq n; a_i \neq b_i)$ – номера роботов, соединенных i-м проводом. Гарантируется, что никакие два провода не соединяют одну и ту же пару роботов.

Также гарантируется, что сумма n по всем наборам входных данных не превосходит 10^5 и сумма m по всем наборам входных данных не превосходит $1.5 \cdot 10^5$.

Формат выходных данных

Для каждого набора входных данных выведите в первой строке через пробел максимальное количество роботов, управление над которыми можно получить, и количество действий k, которое для этого понадобится.

В следующих k строках выведите сами описания действий по одному на каждой строке. Описание действия должно состоять из трех целых чисел t, x и y $(1 \leq t, x, y \leq n; x \neq y)$, означающих, что робот номер t меняет провод (t, x) на провод (t, y).

Если возможных ответов несколько, выведите любой. Обратите внимание, что k при этом обязано быть **наименьшим**, при котором можно подключить максимальное количество роботов.

Пример

Стандартный ввод	**Стандартный вывод**
5 2 0 2 1 1 2 5 3 1 2 2 3 1 3 7 5 1 2 2 3 4 5 5 6 4 6 7 7 1 2 3 4 3 5 3 6 4 5 4 6 5 6	1 0 2 0 4 1 3 2 4 6 2 2 3 4 6 5 7 7 3 1 2 3 6 5 7 6 4 2

Решение:

В этой задаче были необходимы базовые алгоритмы работы с графами, а также идеи жадных алгоритмов: понимание того, какие действия приводят к строго более положительной ситуации, и почему такими действиями можно прийти к глобальному оптимуму.

Первым действием выделим в графе компоненты связности. Каждая компонента связности — это множество вершин, попарно связанных друг с другом путями, то есть последовательностями ребер, имеющих общие концы. Выделим три типа компонент связности:

1. изолированная вершина (вершина, из которой вообще не выходят ребра), которая образует компоненту связности размера 1;

2. дерево размера больше 2; это компонента связности, в которой число ребер на 1 меньше числа вершин, поэтому между любыми двумя вершинами есть единственный путь, и удаление любого ребра ведет к тому, что компонента распадается на два меньших дерева;

3. все оставшиеся – компоненты связности, в которых есть «лишние» ребра, от удаления которых свойство связности всех ее вершин не нарушается.

Две ключевые идеи задачи:

- можно использовать «лишнее» ребро для подключения любой компоненты связности к текущей;

- можно использовать ребро в дереве между листом (вершиной с одним исходящим ребром) и его соседом, чтобы подключить любую другую компоненту, потеряв в процессе этот лист.

Заметим, что при соединении двух компонент их множества лишних ребер объединяются, а в конечном итоге либо лишние ребра останутся, и тогда вообще все роботы подключены к первому (напрямую или косвенно), либо лишних ребер не останется, и тогда компонента связности первого робота будет деревом. Это означает, что в первом случае можно спокойно расходовать лишние ребра, потому что мы все равно сможем подключить всех роботов, а во втором – их можно спокойно расходовать, потому что в итоге они все равно будут израсходованы целиком.

Остается только заметить, что в случае, когда сейчас нет доступных «лишних» ребер, мы можем либо подключить за счет листа компоненту, в которой они есть, и затем подключить все оставшиеся, в которых они есть, либо все оставшиеся компоненты связности – деревья, и тогда остается только обменять листья на оставшиеся деревья размера хотя бы 2, после чего завершить алгоритм – остальные вершины все равно подключить не получилось бы.

Таким образом, полный алгоритм выглядит так.

1. Выделим компоненты связности, это можно сделать с помощью алгоритмов обхода dfs или bfs. При обходе они также строят то, что называется остовным деревом компоненты — множество ребер, образующее дерево. Выделим отдельно дерево и все «лишние» ребра. А для дерева запомним порядок обхода и «родителя» каждой вершины.

2. Оставим отдельно компоненту связности первой вершины.Если она размера 1, то мы ничего не можем сделать, и ответ – «1 0». Иначе отсортируем все остальные компоненты по следующему критерию: сначала по убыванию числа «лишних» ребер, затем по убыванию размера. Тогда изолированные вершины будут в самом конце.

3. По очереди будем подключать эти компоненты в полученном порядке, пока есть возможность. Если у нас есть «лишнее ребро», воспользуемся им. Иначе, если размер компоненты, которую мы хотим подключить, больше 1, потратим лист. Для этого возьмем последнюю в порядке обхода вершину (она точно будет листом в дереве обхода) и потратим ребро из нее в предка. Образуется новая компонента размера 1, ее можно положить в конец списка компонент, но зато мы можем подключить следующую компоненту из списка.

При подключении компоненты объединим ее множество «лишних» ребер с нашим, а также припишем ее порядок обхода в конец к нашему. Действительно, если мы присоединяем новую компоненту, то дальше перед тем, как расходовать листья нашей исходной компоненты, мы можем целиком израсходовать то, что только что подключили, если в этом будет необходимость.

Процесс остановится, когда либо все компоненты будут подключены, либо не останется «лишних» ребер и неподключенных компонент размера больше 1. Весь алгоритм занимает $O(nlogn + m)$ времени из-за обхода графа и сортировки.

Отборочный этап. Первый тур (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балл) [Очень плохая задача]

Дано математическое выражение:

$$BA\ldots AD_X + BA\ldots AD_Y = K_{10}$$

В обоих случаях троеточие заменяет какое-то количество цифр A. Известно, что в каждом из чисел BA…AD по N знаков, число N - чётное, X = 10000000000000, Y = 90000000000005. Определите последние 2 цифры в записи числа K_{10}. Если таких пар цифр несколько, запишите в ответ любую из них.

Ответ: 01

2. Кодирование информации. Количество информации. Элементы комбинаторики (2 балла) [Что будет, если нажать на кнопку?]

При нажатии кнопки происходит одно из 3 независимых событий – A, B или C. Появление одного из событий не оказывает влияния на последующие события. Известно, что если нажать кнопку 5 раз подряд, то последовательность событий ABCBA произойдёт с вероятностью 0,0008, последовательность ABBCC – с вероятностью 0,004, а вероятность не встретить событие B – 0,07776. Определите вероятности наступления событий A, B и C. В ответе укажите через пробел вероятность события A, затем вероятность события B и вероятность события C. Значения вероятностей округлите до первого знака после запятой.

Ответ: 0,1 0,4 0,5

3. Кодирование информации. Количество информации. Кодирование текста (1 балл) [Место под систему]

Вася решил написать программу, решающую системы квадратных уравнений с параметрами. В Васиной школе учат, что уравнение имеет вид $Ax^2 + Bx + C = 0$, где A, B и C – коэффициенты, x - переменная. Запись уравнения не содержит пробельные символы. Коэффициенты и переменная могут быть обозначены любыми из доступных букв. Перед первым коэффициентом не ставится знак.

Вася считает, что для записи уравнений системы могут быть использованы строчные и заглавные символы современного английского (26 букв) и современного греческого (24 буквы) алфавитов, а также знаки из множества { +, =, ², 0 }. Вася сохраняет в память для каждого уравнения полное текстовое представление в описанном выше формате. Определите, сколько места в памяти займёт система из 6 уравнений, если используется равномерное кодирование с минимально возможным количеством бит на один символ? В ответ запишите одно число – искомое количество информации в битах.

Ответ: 420

4. Кодирование информации. Объем данных (2 балла)

[Видеодневник]

Григорий разрабатывает собственное приложение для видеозаметок. Он собирается ежедневно сохранять от 1 до 5 видеофрагментов длиной от 10 до 30 секунд в Full HD (1920x1080 пикселей) палитрой в 65536 цветов и частотой 30 кадров/секунду с двухканальной аудиодорожкой с частотой дискретизации 20207 Гц и 32768 уровнями квантования. Для каждого видеофрагмента сохраняется также 960 Кбайт метаинформации. Сжатие данных не используется.

Определите:

1. Минимальное возможное число дней в дневнике, если дневник занимает не менее 5000 Мб?
2. Максимальное возможное число дней в дневнике, если дневник занимает не более 5000 Мб?

В ответ запишите через пробел 2 числа – ответы на первый и второй вопросы соответственно.

Ответ: 1 4

5. Основы логики. Анализ логических функций (2 балла)

[Степень похожести]

Даны логические функции:

1. A and B or not A and C
2. A and C or not B and C
3. C or A and B
4. C or not A or not B
5. C or not (B or not A)
6. not (B or not C) or A
7. B or not (C or not A)
8. (A or B) and (B or C)

Назовём степенью похожести множества логических функций число различных наборов переменных, при которых все логические функции этого набора принимают одинаковое значение. Например, для эквивалентных функций степень похожести будет равна числу различных наборов переменных, для функций A и NOT(A) – нулю, а для функций A OR B и A AND B – двум. Среди приведённых логических функций от трёх переменных найдите подмножество мощности больше 1 с максимально возможной степенью похожести. В ответ укажите сначала степень похожести в найденном подмножестве, а затем через пробел номера функций в искомом подмножестве в порядке возрастания. В случае, если таких подмножеств несколько, укажите то, в котором сумма номеров элементов больше.

Пример записи ответа: 7 2 4 6 8

Ответ: 7 1 3

6. Основы логики. Упрощение логического выражения (1 балл)

[Бей крота]

Упростите логическое выражение или укажите его результат (при его однозначности).

$$A \wedge B \wedge C \vee A \wedge \bar{B} \wedge C \vee A \wedge B \wedge \bar{C} \vee A \wedge \bar{B} \wedge \bar{C} \vee \bar{A} \wedge \bar{B} \wedge C \vee \bar{A} \wedge \bar{B} \wedge \bar{C}$$

Результат упрощения может содержать только операнды, операции инверсии, конъюнкции и дизъюнкции и не должен содержать скобок. Будем считать выражение упрощенным, если не существует эквивалентного ему выражения, содержащего меньшее количество логических операций и скобок.

Комментарий по вводу ответа: операнды вводятся большими латинскими буквами; логические операции обозначаются, соответственно, как not, and и or.

При однозначном ответе – истинный ответ обозначается как 1, а ложный как 0.

Пример записи ответа: A or B and not C

Ответ: A or not B || not B or A

7. Основы логики. Синтез логического выражения (2 балла)

[Чего-то не хватает]

Дана логическая функция от трёх переменных:

$$F(A,B,C) = (A \rightarrow (B \rightarrow C))\ XOR\ G(A,B,C)$$

В данном выражении G(A, B, C) – логическая функция от трёх переменных, → - операция импликации, XOR – операция «исключающее или». Ниже приведена таблица истинности функции F:

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Используя данную таблицу истинности, восстановите и запишите в ответ используя минимальное число операций функцию G(A, B, C).

Пример записи ответа: (A or not B) and C

Примечание: переменные вводятся большими латинскими буквами; логические операции обозначаются, соответственно, not, and и or. Скобки используются только для изменения порядка выполнения операций. Если порядок выполнения операций очевиден из их приоритетов, дополнительное использование скобок считается ошибкой. Пробелы ставятся между логическими операциями и переменными.

Ответ: A or B and C || A or C and B || B and C or A || C and B or A

8. Алгоритмизация и программирование. Формальный исполнитель (3 балла) [Очень странные часы]

Некоторые часы имеют 12 делений и всего одну стрелку. Вера придумала игру с этими часами. Она переводит стрелку часов по следующим правилам:

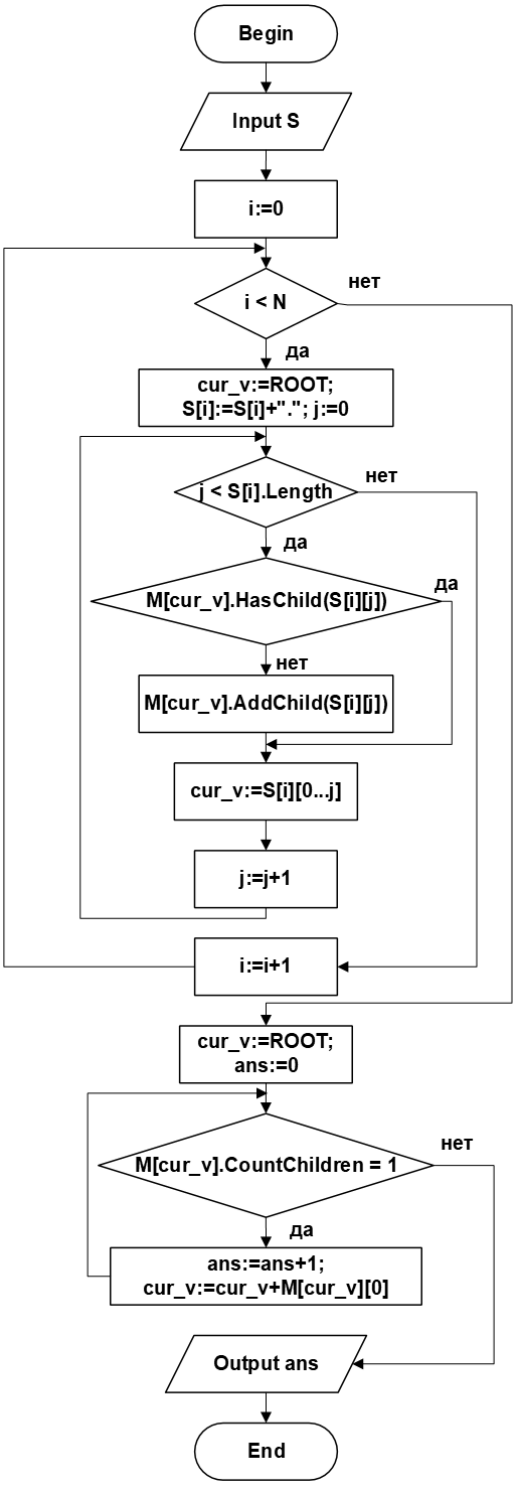
1. Каждый ход стрелка переводится в направлении противоположном прошлому.
2. Стрелка переводится на X делений.
3. X увеличивается на 1. Если после этого X стало равно 16, X становится равным 1.
4. В момент начала игры стрелка стоит на делении 1, X=1, на первом ходу стрелка переводится вперёд, в направлении по часовой стрелке.

Сколько раз стрелка пересекла отметку 16 или остановилась на ней после перевода, если Вера перевела часы 1000 раз? Начало движения на отметке 16 не является пересечением отметки 16.

Ответ: 470

9. Алгоритмизация и программирование. Блок-схема, массивы, обратная задача (3 балла) [Тестовые данные повреждены]

Дана блок-схема алгоритма:



В данной блок-схеме S – массив строк разных длин, N – число строк в массиве S, M – ассоциативный массив, ключом является строка, значением – набор символов, ROOT – константа, обозначающая начало строки.

S[i][0…j] означает, что у i-ой строки в массиве S берётся подстрока от 0-го до j-го символа включительно. Например, если S[i] = “abcdef”, то обозначение S[i][0…0] будет соответствовать строке “a”, а S[i][0…3] будет соответствовать строке “abcd”.

Метод M[i].HasChild(c) проверяет, есть ли в ассоциативном массиве M в наборе значений, соответствующем ключу i, символ c.

Метод M[i].AddChild(c) добавляет в ассоциативном массиве M в набор значений, соответствующий ключу i, символ c.

Свойство M[i].CountChildren содержит информацию о числе символов в наборе значений, соответствующем ключу i в ассоциативном массиве M.

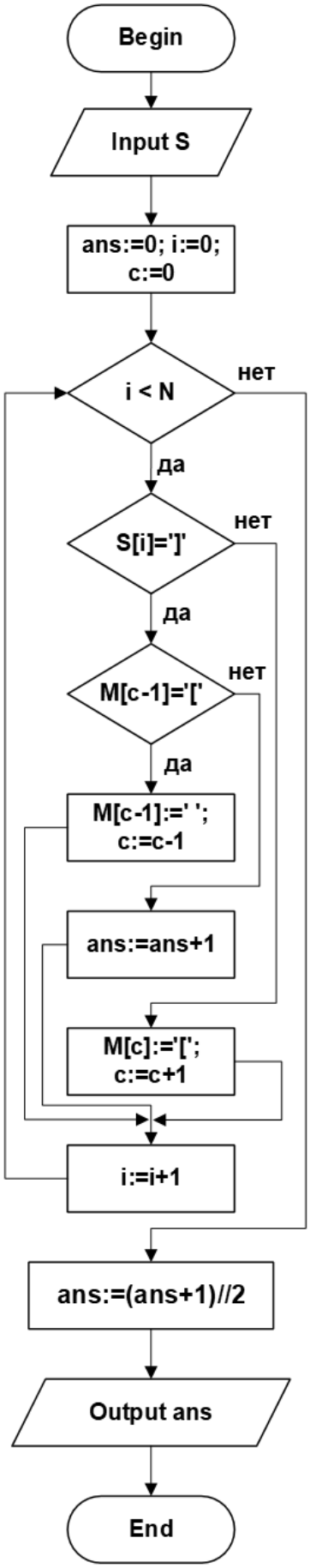
В результате работы алгоритма была выведено число 10.

Вам удалось узнать, что на вход был подан массив строк { **abbcaabbccabcabc, abbcaabbcccabcabc, abbcaabbccaabcabc, abbcaabbccbacabc, abbcaabbcabcabc, abbcaabbccabcacb** }, однако одна из строк была добавлена уже после окончания работы алгоритма. Определите эту строку.

Ответ: abbcaabbcabcabc

10. Алгоритмизация и программирование. Блок-схема, обратная задача (2 балла) [Особенные четырёхзначные]

Дана блок-схема алгоритма:



В данной блок-схеме: S – строка длины N из символов ‘[‘ и ‘]’, M – массив длины не более N, заполненный пробельными символами ' '. Оператор // вычисляет частное от целочисленного деления. Если некоторого элемента, например, M[-10] нет или не может существовать, то его равенство чему-либо автоматически невозможно.

Определите, какое число было выведено на экран, если на вход была дана строка]]]]][[[[[. Если данная строка не может быть корректно обработана данным алгоритмом, укажите в ответ NULL.

Ответ: 3

Отборочный этап. Второй тур (приведен один из вариантов заданий)

1. Электронные таблицы. Адресация ячеек и вычисления (1 балл) [Узнать знакомого]

	A	B	C	D	E	F
1			660	4914	2340	9009
2			220	14	20	77
3	660	220	660	6	60	33
4	4914	14	6	4914	234	819
5	2340	20	60	234	2340	117
6	9009	77	33	819	117	9009

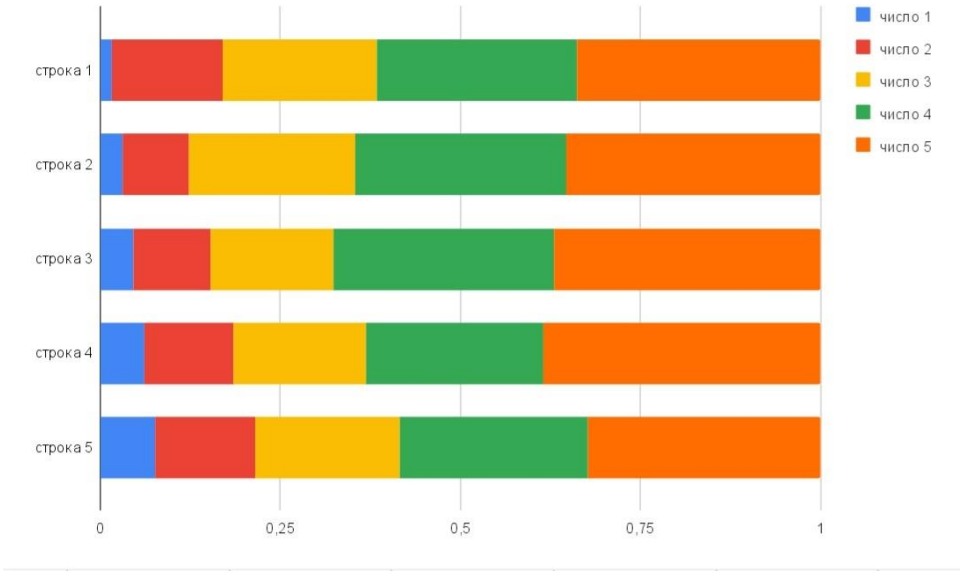
В ячейку таблицы B2 поместили некоторую стандартную функцию от двух аргументов, после чего скопировали её во все ячейки диапазона B2:F6. В ячейки B1:F1 поместили 5 различных чисел, после чего скопировали их в ячейки A2:A6 (число из B1 было скопировано в A2, из C1 в A3 и так далее). После этого таблица приняла следующий вид:

Определите минимальное возможное натуральное число в ячейке B1.

Ответ: 1540

2. Электронные таблицы. Графики и диаграммы (2 балла) [От 1 до 25]

Таблицу 5x5 заполнили числами от 1 до 25 без повторов. После этого по таблице составили нормированную линейчатую диаграмму с накоплением:



Определите значение в центральной ячейке таблицы, если известно, что во всех строках таблицы получились равные суммы.

Ответ: 11

3. Сортировка и фильтрация данных (2 балла) [Многочисленные навыки]

В некотором симуляторе колонии выбирается группа поселенцев из имеющихся кандидатов. У каждого поселенца есть 12 навыков:

1. Ближний бой
2. Дальний бой
3. Строительство
4. Горное дело

5. Кулинария
6. Растениеводство
7. Животноводство
8. Ремесло
9. Искусство
10. Медицина
11. Общение
12. Умственный труд

	Б. бой	Д. бой	Стр-во	Гор. д.	Кулин.	Раст-во	Жив-во	Ремес.	Иск-во	Мед.	Общ.	Ум. труд
Поселенец 0	2	10	0	8	4	10	2	13	7	14	5	3
Поселенец 1	9	12	3	6	12	8	5	8	7	8	13	1
Поселенец 2												
Поселенец 3	13	2	8	1	6	16	4	4	8	8	13	5
Поселенец 4	13	6	1	3	11	14	1	12	3	1	5	3
Поселенец 5	11	1	8	9	4	13	1	10	10	1	2	6
Поселенец 6	10	6	14	2	6	0	2	1	3	10	12	9
Поселенец 7	3	5	2	1	9	7	4	11	13	0	7	14
Поселенец 8	8	9	4	4	12	4	9	9	2	1	14	5
Поселенец 9	0	11	0	9	3	6	10	12	10	0	14	7
Группа	13	11	6	9	11	14	10	12	10	15	16	13

Владение каждым из навыков оценивается числом от 0 до 20, где 0 – совсем не владеет, 20 – владеет в совершенстве. Владение тем или иным навыком группы вычисляется как максимальное из значений у поселенцев этой группы. В качестве кандидатов были представлены 10 поселенцев, таблица их способностей приведена ниже:

В состав группы из 3 поселенцев вошли поселенцы 2, 4 и 9. Известно, что поселенцы выбирались таким образом, чтобы сумма величин навыков группы из 3 поселенцев была максимальна. Также известно, что полученная сумма равнялась 140.

Определите число возможных значений строки навыков у поселенца 2.

Ответ: 475675200

4. Поиск и фильтрация данных (3 балла) [Список из нескольких строк]

Васе попал в руки список строк:

1. mwkNVWCVC2mL34C
2. mnnDdNXvVd2BDZa
3. Jminh6maTT6uHq3
4. fMPbgkkAR3RYF22
5. WCaHmy8wMUrS5Aa
6. jN3FrLgN49KaY3J
7. jwQ2CA5afhEczYm
8. x5u9jGgkecD5i2q
9. uzP2MFs4XK49wdT
10. sLHgN8BpsnsM3Bp
11. XDt9giUj2vnQci4
12. BE4n6WShNGi8aHN
13. TXDhGG3kJZgVAgD
14. PwgWYrJc83FBWxP
15. xuTYLbdNea6mS7B
16. GdaXEiQCXHS7Xn9
17. hYXDqbcvjFM7E9d
18. xfwUrCa2622CEzV
19. wW7aZkA2d9iWWsD
20. VN4fJBNWJAXn9vP

Вася посчитал, сколько существует пар различных букв, таких, что выбрав все слова из списка, где встречаются обе эти буквы, и есть хотя бы одна подстрока, в которой первая буква пары идёт раньше второй, получится список из более чем N строк. Определите минимальное возможное N, если известно, что Вася насчитал менее 40 пар.

Примечание: например, если парой букв будут a и b, то строки cacb и cbab Васе подойдут, а строки cbba и cbbb – нет. Регистр букв не имеет значения. Рассматриваются только пары букв, пары цифр или пары "цифра-буква" не рассматриваются. Пары ab и ba - разные, т.к. важен порядок, в котором буквы встречаются в строке.

Ответ: 4

5. Телекоммуникационные технологии (3 балла) [Я с тобой больше не разговариваю]

По защищенному каналу взаимодействуют устройства А и Б.

Устройство А может выполнять следующие типы операций:

• Проверка пакета перед отправкой (скорость операции - 1 Мб/с),
• Отправка пакета (скорость операции - 4 Мб/с),
• Проверка наличия шифрования на пакете (операция выполняется за 1 с),
• Дешифровка пакета (операция выполняется со скоростью 2 Мб/с),
• Проверка совпадения дешифрованного пакета с одним из последних 5 отправленных (операция является мгновенной),
• Отправка устройству Б уведомления о прекращении взаимодействия.(операция является мгновенной),
• Приём уведомления о недоставке пакета (операция является мгновенной).

Устройство Б может выполнять следующие типы операций:
• Выбор пакета из 5 последних полученных (операция является мгновенной),
• Приём пакета от устройства А (операция является мгновенной),
• Приём уведомления о прекращении взаимодействия от устройства А (операция является мгновенной),
• Шифровка пакета (операция выполняется со скоростью 1.5 Мб/с),
• Отправка пакета устройству А (операция выполняется со скоростью 4 Мб/с),
• Уведомление устройства А о недоставке пакета (операция является мгновенной).

Передача пакетов может происходить одновременно в обоих направлениях. Пакет доставляется целиком по прошествии t секунд, где t - размер пакета(в Мб)/скорость передачи (в Мб в секунду). Для обоих устройств отправка пакета является активным процессом.

При передаче информации по защищенному каналу с устройства А на устройство Б используется алгоритм проверки безопасности подключения: после принятия каждых 5 пакетов, устройство Б берёт один из них, шифрует, затем осуществляет передачу зашифрованного пакета устройству А.

Перед отправкой каждого пакета устройство А проверяет этот пакет.

Если во время проверки некоторого пакета X перед отправкой или отправки пакета X устройству А приходит пакет от Б, текущий активный процесс прерывается, и устройство А начинает проверку пакета от Б. Устройство А проверяет, что пакет зашифрован и далее дешифрует пакет.

В случае, если пакет не был зашифрован или если полученный в результате дешифровки пакет не является одним из 5 последних отправленных пакетов, устройство А оповещает устройство Б о прекращении соединения и прекращает все взаимодействия с устройством Б.

Если проверка проходит успешно, А возвращается к передаче пакетов и начинает заново выполнять проверку перед отправкой пакета X, вне зависимости от того, какая операция над пакетом X была прервана. То есть, если для пакета не были выполнены и проверка, и отправка, устройство А считает, что ещё не взаимодействовало с этим пакетом.

Если пакет от устройства А приходит в тот момент, когда устройство Б осуществляет шифрование контрольного пакета или передачу зашифрованного пакета, оно не принимает пакет и моментально уведомляет об этом А. Уведомление о недоставке пакета устройство А получит моментально. Если пакет не доставлен, устройство А пытается отправить пакет заново: сначала проводится проверка пакета, затем отправка, до тех пор, пока пакет не будет доставлен или не будет остановлена передача.

Определите, сколько секунд заняла отправка 756 пакетов, если устройство А последовательно отправляло пакеты размером 12 и 48 Мб, а устройство Б выбирало в качестве контрольного всегда 4-й пакет.

Ответ: 34944

6. Операционные системы (2 балла) [Бюджетные распределенные вычисления]

Кеша заинтересовался распределёнными вычислениями и написал программу для тестирования различных кластеров (наборов процессоров). Кешина программа довольно проста - она добавляет в очередь 10000 операций по 6 секунд каждая и запускает задачи из этой очереди как только у одного из процессоров появляется свободный поток. Запущенная задача из очереди убирается. Программа работает до тех пор, пока очередь не опустеет. Программа не дожидается окончания работы отправленных задач, работа завершается сразу же, как только в очереди не остаётся ни одной задачи. Отправку задач Кеша для простоты вычислений считает мгновенной.

Кеша рассматривает 3 вида процессоров:

A - 8 ядер, 2 потока на каждом ядре, стоимость - 500 денег за один процессор.

B - 16 ядер, 2 потока на каждом ядре, стоимость - 700 денег за один процессор.

C - 4 ядра, 1 поток на каждом ядре, стоимость - 200 денег за один процессор.

Кеша хочет собрать кластер, купив некоторое количество процессоров того или иного типа для запуска на них своей программы. Определите, какое минимальное время работы программы Кеша может достичь, если он хочет, чтобы все используемые процессоры в сумме стоили не более 5000 денег. В качестве ответа укажите через пробел 2 числа – минимальное возможное число секунд и необходимое для этого количество денег.

Примечание: кластер может состоять из процессоров разных типов, а может и из процессоров одного типа.

Ответ: 264 4900

7. Технологии программирования (3 балла)

[Морской бой]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	4 секунды
Ограничение по памяти	256 мегабайт

Вам предстоит проэмулировать игру в Морской бой. В рамках данной задачи давайте считать, что игроки честные, корректно расставляют корабли на поле, и никогда не обманывают противника.

Игра в Морской бой проходит на клетчатом поле размером $n \times m$ (n строк и m столбцов). Строки занумерованы от 1 до n **снизу вверх**, столбцы — буквами слева направо. Первые 26 столбцов нумеруются от 'A' до 'Z', а если столбцов больше, то следующие 26 нумеруются от 'AA' до 'AZ'.

У каждого из двух игроков есть свое неизвестное противнику поле, на котором он расставляет корабли. Корабли представляют из себя полосы $1 \times x$ или $x \times 1$, где x от 1 до L, и расставляются на поле так, что никакие два корабля не соприкасаются по стороне, то есть если две занятые клетки имеют общую сторону, они принадлежат одному кораблю. Соприкосновение по углу разрешается. О количестве кораблей каждого типа игроки договариваются заранее, но строгих ограничений нет.

Вам дана изначальная расстановка кораблей на поле, а также последовательность ходов игроков. В течение хода игрок совершает один «выстрел»: игрок называет клетку противника, после чего противник сообщает вердикт: «мимо», если клетка свободная, «попал», если клетка занята, и «убил», если это было попадание в последнюю клетку соответствующего корабля.

В отличие от стандартных правил,

- игроки ходят строго по очереди вне зависимости от вердикта их последнего хода;
- у каждого игрока есть «бонус» — каждый игрок может в начале игры пометить k клеток кораблей на своем поле как «усиленные» — тогда по ним потребуется попасть дважды, чтобы стоящий на них корабль считался потопленным.

Проанализируйте запись ходов игры и выведите вердикт каждого хода и информацию о победителе, если игра на каком-то ходу завершается (все корабли одного из игроков потоплены).

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода находится одно целое число tt — количество наборов входных данных ($1 \le t \le 50$). Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит четыре целых числа n, m, L и k — размеры поля, максимальную длину корабля и число бонусов у каждого игрока ($4 \le n,m \le 50$; $1 \le L \le 10$; $1 \le k \le 30$).

Следующие n строк содержат по m символов каждая и задают расстановку кораблей первого игрока. Каждый символ — это либо 'X' (клетка корабля), либо '#' (усиленная клетка корабля), либо '.' (пустая клетка). Затем следуют еще n строк, задающие поле второго игрока в том же формате.

Гарантируется, что расстановка корректная и соответствует всем правилам игры.

Следующая строка содержит одно целое число q — количество ходов в игре ($1 \le q \le 2nm$). В i-й из следующих q строк задан ход игрока в формате «FIRE ri ci», где ri — номер строки (в нумерации снизу вверх), а ci — номер столбца, записанный одной или двумя буквами от 'A' до 'Z' в соответствии с описанием нумерации из условия ($1 \le ri \le n$; 'A' $\le$ ci $\le$ 'AX').

Формат выходных данных

После каждого хода игры выведите на отдельной строке одну из следующих строк:

- «MISSED», если ход был сделан мимо (или данная клетка уже потоплена, то есть по ней было одно попадание в случае обычной клетки корабля или два попадания в случае усиленной);
- «HIT», если игрок текущим ходом задел корабль противника;
- «SUNK», если игрок текущим ходом потопил корабль противника;
- «WIN p», если игрок номер p ($1 \le p \le 2$) текущим ходом выиграл.

После того, как какой-то игрок выиграл, если во входных данных остались еще какие-то ходы, их следует проигнорировать и не выводить ничего (до следующего набора входных данных).

Пример

Стандартный ввод	Стандартный вывод
1 4 4 3 1 ....X X... .X#X XXX. ...X	SUNK SUNK MISSED HIT MISSED HIT HIT HIT

Стандартный ввод	Стандартный вывод
.... .#.. 12 FIRE 3 D FIRE 2 A FIRE 3 C FIRE 1 C FIRE 4 D FIRE 1 B FIRE 4 A FIRE 1 D FIRE 2 D FIRE 1 C FIRE 4 B FIRE 3 D	MISSED SUNK HIT WIN 2

8. Технологии программирования (4 балла)
[Морской бой]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	1.5 секунды
Ограничение по памяти	256 мегабайт

У Паши есть три массива a, b и c. Каждый из них имеет длину n.

Паша любит заниматься комбинаторикой, но поскольку он пока не очень взрослый, его «комбинаторика» заключается в комбинировании нескольких разных объектов в один. В данном случае Паша собирает массив d следующим образом.

1. Сначала он выбирает два индекса i, j ($1<i<j\le n$);
2. Затем он берет префикс массива a длины $i-1$ (то есть все его элементы на позициях от 1 до $i-1$), обозначим его как $a_1 \dots a_{i-1}$, отрезок массива b с индексами от i до $j-1$ ($b_i \dots b_{j-1}$), и суффикс массива c длины $n-j+1$ — элементы с j-го по n-й ($c_j \dots c_n$);
3. Три массива, полученные в предыдущем пункте, склеиваются в том же порядке: полученный массив и будет массивом d.

Обратите внимание, что, исходя из ограничений на i и j, выбранные подмассивы массивов a, b и c **не могут быть пустыми**.

Паша задался вопросом — какие индексы i и j нужно выбрать, чтобы максимизировать сумму элементов в массиве d? Подскажите ему ответ на этот вопрос.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода находится одно целое число t — количество наборов входных данных ($1\le t\le 10^4$). Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит целое число n — длину массивов ($3\le n\le 10^5$).

Вторая, третья и четвертая строки содержат по n целых чисел — элементы массивов a, b и c соответственно ($1\le a_i,b_i,c_i \le 10^9$ для всех i).

Гарантируется, что сумма n по всем наборам входных данных не превосходит 10^5.

Формат выходных данных

Для каждого набора входных данных в отдельной строке выведите три числа — индексы i и j, которые нужно выбрать, чтобы достичь максимальной суммы элементов массива d, и сумму элементов массива d в таком случае. Если подходящих ответов несколько, выведите любой из них.

Пример

Стандартный ввод	Стандартный вывод
3 5 1 2 3 4 5 7 4 2 11 1 5 4 3 2 1 3	2 5 19 2 3 15 3 6 8

Стандартный ввод	Стандартный вывод
1 2 3 4 5 6 7 8 9 6 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2	

Задания для 5–8 класса

Заключительный этап (приведен один из вариантов заданий)

1. Кодирование информации, кодирование звуковой информации (1 балл)

[Тяжело записывать оркестр]

Мила играет в духовом оркестре. Оркестр записал выступление, создав отдельные аудиозаписи для каждого инструмента. Чтобы прослушать композицию в исполнении всего оркестра, нужно запустить все аудиофайлы одновременно через аудиоредактор (записи всех инструментов имеют одинаковую продолжительность).

Мила хочет отправить запись выступления подруге в социальной сети, однако выяснилось, что в этой социальной сети нельзя отправить набор файлов, общий информационный объем которых превышает 2 Гб.

Мила решила уменьшить число звуковых файлов. С помощью аудиоредактора она может превратить два одноканальных аудиофайла в один одноканальный, в котором звучание инструментов исходных файлов просто наложится друг на друга. Все характеристики выходного файла (такие как частота, длительность, глубина кодирования) совпадают с соответствующими характеристиками исходных файлов. Сразу после выполнения этого действия исходные два файла заменяются одним, полученным в результате выполнения этой операции.

Какое минимальное количество объединений потребуется сделать Миле, чтобы она смогла отправить получившийся набор файлов в социальной сети, если в ее оркестре 55 музыкантов, каждый инструмент записан в одноканальном режиме при частоте 40кГц и глубине кодирования 16 бит, а композиция длится 8 минут 32 секунды?

В ответ запишите одно целое число – минимальное количество операций объединения.

Ответ: 3

Решение:

Подсчитаем информационный объем записи одного инструмента.

Применим формулу $\boldsymbol{I = D \cdot t \cdot i \cdot k}$.

$$\boldsymbol{I} = \mathbf{40\,000}\ (\text{Гц}) \cdot \mathbf{512}\ (\text{секунд}) \cdot \mathbf{16}\ (\text{бит}) \cdot \mathbf{1}\ (\text{число каналов}) = \mathbf{327\,680\,000}\ (\text{бит}) = \mathbf{40\,000}\ \text{Кб}$$

Рассчитаем информационный объем 55 таких аудио файлов: $\mathbf{40\,000 \cdot 55 = 2\,200\,000}$ Кб

Рассчитаем, сколько Кб может занять набор файлов, допустимый для передачи в социальной сети.

$$\mathbf{2 \cdot 1024 \cdot 1024 = 2\,097\,152}\ \text{Кб}$$

Заметим, что одна операция объединения удаляет ровно один файл из исходного набора фалов. Это значит, что каждая операция объединения уменьшает общий информационный объем на 40 000 Кб.

Теперь составим уравнение, где x – искомое число операций объединения.

$$\mathbf{2\,200\,000 - 40\,000 \cdot} \boldsymbol{x} \mathbf{< 2\,097\,152}$$

Найдем минимальный x, удовлетворяющий условию – это 3.

2. Алгоритмизация и программирование (2 балла)

[Закон Амдала]

Выполнение программы можно ускорить, если выполнять разные части программы параллельно на разных ядрах компьютера. Обозначим как ускорение S показатель того, во сколько раз программа выполняется быстрее при параллельном вычислении на нескольких ядрах по сравнению с последовательным (одноядерным) выполнением. Максимальное ускорение, которое можно получить, распараллелив вычисления наиболее эффективно, выражается в законе Амдала:

$\boldsymbol{S} = \frac{\mathbf{1}}{(\mathbf{1}-\boldsymbol{P})+\left(\frac{\boldsymbol{P}}{\boldsymbol{N}}\right)}$, где:

$\boldsymbol{P}$ – это величина, рассчитывающаяся из выражения $\mathbf{1} - \boldsymbol{Q} = \boldsymbol{P}$, где $\boldsymbol{Q}$ – это отношение доли кода, который нельзя выполнять одновременно с каким-либо другим кодом этой программы, ко всему коду программы.

$\boldsymbol{N}$ – число ядер, которые могут быть задействованы для выполнения программы.

Король волшебного королевства "Информат", где магические заклинания записываются как программы, пригласил талантливого инженера Виктора, чтобы модернизировать свой одноядерный компьютер.

Виктор выяснил, что $\mathbf{80}\%$ всех магических заклинаний, выполняемых королем, могут выполняться параллельно на нескольких ядрах. Оставшиеся $\mathbf{20}\%$ заклинаний не могут выполняться одновременно ни между собой, ни вместе с другими заклинаниями, так как это приведет к побочным эффектам.

Виктор предложил добавить в компьютер $\mathbf{4}$ новых ядра и настроить работу устройства так, чтобы на новых ядрах выполнялись только заклинания, которые могут выполняться параллельно друг другу, а на исходном ядре - только заклинания, которые нельзя выполнять одновременно с чем-либо (которые выполняются тогда и только тогда, когда ни на каком из других ядер не выполняются никакие заклинания).

Король задался вопросом: во сколько раз быстрее будет работать королевский компьютер после модернизации? Помогите королю найти ответ.

Ответ округлите до $\mathbf{1}$ знака после запятой, в качестве разделителя используйте запятую. Например, если компьютер станет работать в 1,1 раз быстрее, то в ответ следует указать "$\mathbf{1,1}$".

Ответ: 2,5

Решение:

Доля кода, которую можно распараллелить:

$\boldsymbol{P} = \mathbf{0,8}$

Следовательно доля кода, которая исполняется последовательно:

$\mathbf{1} - \boldsymbol{P} = \mathbf{0,2}$

У нас появляется $\mathbf{4}$ новых ядра для параллельной части. При этом последовательная часть выполняется только тогда, когда никакие параллельные задачи не выполняются, а параллельная часть распределяется равномерно между $\mathbf{4}$ ядрами.

Тогда работу системы можно визуализировать следующим образом (столбцы представляют собой ядра системы, строки – отрезки времени, закрашенное пересечение отрезка времени и ядра означает, что ядро в этот отрезок времени работало). Процесс работы показан сверху вниз.

Следует отметить, что хотя бы одно ядро всегда простаивает. Очевидно, эффективность системы Виктора эквивалентна эффективности вот такой системы на 4-х ядрах, которая, в отличие от системы Виктора, распараллеливает вычисления наиболее эффективно, а значит к ней уже можно применить закон Амдала.

Подставим значение $\boldsymbol{N = 4}$ (ядер в системе) в формулу, данную в задаче:

$$\boldsymbol{S = \frac{1}{(1-P) + \left(\frac{P}{N}\right)} = \frac{1}{(1-0,8) + \left(\frac{0,8}{4}\right)} = 2,5}$$

Пояснение к $\frac{P}{N}$:

Скорость исполнения параллельной части выполняется на $\mathbf{4}$ ядрах, поэтому затраченное время сокращается в $\mathbf{4}$ раза $\frac{0,8}{4} = \mathbf{0,2} \cdot \frac{P}{N}$

3. Кодирование информации. Системы счисления (2 балла)

[Время]

В древней хронике время отсчитывалось в минутах от начала дня. Количество минут, прошедших с начала дня, записывали в виде числа в системе счисления с основанием $\boldsymbol{b}$.

Известно, что когда с начала дня прошло ровно $\mathbf{64}$ минуты, текущее время можно было записать как двухзначное число AB, где $\boldsymbol{A}$ и $\boldsymbol{B}$ – это цифры системы счисления с основанием b. Также известно, что если по отдельности перевести A и B (рассматривая их как самостоятельные числа) в десятичную систему счисления, то $\boldsymbol{A + B = 9}$.

Примечание: числа $\mathbf{64}$, $\mathbf{9}$ записаны в десятичной системе.

Какое минимальное основание $\boldsymbol{b}$ может быть у этой системы счисления? В ответ укажите единственное число – $\boldsymbol{b}$.

Ответ запишите в десятичной системе.

Ответ: 12

Решение:

Запишем условие задачи в виде уравнений. Время, равное 64 минутам записано в системе счисления с основанием $\boldsymbol{b}$ записывается как двузначное число $\boldsymbol{AB}$, а его значение в десятичной системе счисления:

$\boldsymbol{A \cdot b + B = 64}$

Также известно, что при переводе цифр A и B в десятичную систему их сумма равна $\mathbf{9}$: $\boldsymbol{A + B = 9}$

Мы можем выразить $\boldsymbol{B}$ через $\boldsymbol{A}$: $\boldsymbol{B = 9 - A}$

Подставим выражение для B в первое уравнение:

$\boldsymbol{A \cdot b + (9 - A) = 64}$

$\boldsymbol{A \cdot b - A + 9 = 64}$

$\boldsymbol{A(b - 1) + 9 = 64}$

$\boldsymbol{A(b - 1) = 55}$

Найдем все возможные разложения числа $\mathbf{55}$ на два целых числа:

$\mathbf{1 \cdot 55}$ и $\mathbf{11 \cdot 5}$ (все остальные варианты не выполняют $\boldsymbol{A < b}$)

Если $\boldsymbol{A = 1}$, то $\boldsymbol{b - 1 = 55}$ –> $\boldsymbol{b = 56}$. Тогда $\boldsymbol{B = 9 - 1 = 8}$, проверяем: $\mathbf{1 \cdot 56 + 8 = 56 + 8 = 64}$

Если $\boldsymbol{A = 5}$, то $\boldsymbol{b - 1 = 11}$ –> $\boldsymbol{b = 12}$. Тогда $\boldsymbol{B = 9 - 5 = 4}$, проверяем: $\mathbf{5 \cdot 12 + 4 = 60 + 4 = 64}$

Оба варианта подходят, но по условию нам нужно записать минимальное значение из всех возможных.

4. Основы логики (2 балла)

[Матроскин и логический преобразователь]

Кот Матроскин придумал свой логический преобразователь, которому на вход подается функция от 4 переменных. Работает он следующим образом:

1. Строится таблица истинности данной функции.
2. В переменную A кладется значение функции на наборе из всех нулей.
3. В переменную B кладется значение функции на наборе из всех единиц.
4. Переменная R вычисляется по следующей формуле:

   R=(НЕ(A) ИЛИ НЕ(B)) И (A ИЛИ B)
5. Результатом работы логического преобразователя является значение переменной R.

Кот Матроскин хочет узнать, сколько существует таких различных функций от 4 переменных, для которых преобразователь вернет ИСТИНУ.

Функции называются различными, если их таблицы истинности различаются хотя бы одной строчкой.

В ответе укажите целое число.

Ответ: 32768

Решение:

Заметим, что формула, по которой вычисляется переменная R равносильна Исключающему ИЛИ. Из этого можно сделать вывод, что логический преобразователь будет возвращать ИСТИНУ только в том случае, если переданная на вход функция возвращается ИСТИНУ либо на наборе из нулей, либо на наборе из единиц. Если же функция, переданная на вход, на обоих рассматриваемых наборах возвращает ЛОЖЬ или на обоих рассматриваемых наборах возвращает ИСТИНУ – то логический преобразователь вернет нам ЛОЖЬ.

Всего функций от 4 переменных: $\mathbf{2^{2^4} = 2^{16}}$. Из них функций, возвращающих ИСТИНУ на нулевом наборе: 2^{15}. По условию задачи, на наборе из единиц такие функции должны возвращать ЛОЖЬ. Следовательно, всего таких функций будет 2^{14}.

Аналогично рассматриваем случай для функций, которые на нулевом наборе возвращают ЛОЖЬ. Таких тоже будет 2^{14}. Тогда всего подходящих нам функций $\mathbf{2 \cdot 2^{14} = 2^{15} = 32768}$

5. Основы комбинаторики (1 балл) [По ЕГЭ, так по ЕГЭ]

Александр Санечкин, так и не выиграв олимпиаду, решил сдавать ЕГЭ. На экзамене ему предложено 11 вопросов, каждый с четырьмя вариантами ответа: A, B, C и D. Александр считает, что вариант B зачастую оказывается правильным, поэтому он решил действовать по следующему плану:

Александр поставит ответ B ровно в 5 вопросах, при этом никакие два вопроса с ответом B не должны идти подряд (то есть между любыми двумя вопросами с ответом B должен быть хотя бы один вопрос с другим ответом).

В остальных 6 вопросах он будет выбирать ответы только из вариантов A, C или D, и при этом решил, что все три этих варианта должны встречаться хотя бы один раз.

Сколько существует способов заполнить бланк ответов, удовлетворив всем условиям его плана?

Ответ: 11340

Решение:

Для начала выпишем все условия, о которых говорится в задаче:

1. Ровно 5 ответов должны быть вариантом B

2. Никакие два ответа B не идут подряд, то есть между любыми двумя B должен стоять хотя бы одна "не B"

3. Все ответы "не B" Александр выбирает из трёх вариантов (A, C, D), но есть условие, что каждый из вариантов должен встретиться хотя бы один раз.

Посчитаем сколько есть способов расставить 5 букв B на 11 позиций, чтобы две B не шли подряд:

Чтобы было удобнее будем расставлять 6 "не B", они создают 7 промежутков, где могут стоять B.

Выбираем из 7 промежутков 5: $\binom{7}{5}$ = 21

Получается, что у нас есть 21 способ расставить 5 букв B так, чтобы никакие две не шли подряд.

Теперь расставляем остальные варианты ответов:

Всего способов расставить A, C и D на 6 позиций равняется $3^6 = 729$

Вычитаем случаи, где не появляется любой из вариантов ответа:

A: $2^6 = 64$

C: $2^6 = 64$

D:$2^6 = 64$

Теперь нам нужно вернуть случаи, когда не появляется сразу два ответ (к примеру, ни A, ни C), следовательно их нужно вернуть:

A, C: $1^6 = 1$ способ

A, D: $1^6 = 1$ способ

C, D: $1^6 = 1$ способ

Общее число расстановок ответов A, C, D: $3^6 – 3 \cdot 2^6 + 3 = 729 – 192 + 3 = 540$

Умножаем количество способов расстановки A, C, D, на количество способов расстановки B: $21 \cdot 540 = 11340$

6. Информационное моделирование. Моделирование на строке (3 балла) [Смещения]

Андрей придумал свой собственный способ шифрования слов. Шифрование состоит из двух этапов, в каждом из которых над всеми буквами производился ряд смещений их на некоторое число.

Операция смещения буквы на число X подразумевает замену этой буквы на букву, находящуюся на X позиций правее в русском алфавите. Например, смещение буквы ‘А’ на 5 дает букву ‘Е’. Алфавит считается зацикленным, т. е. после буквы ‘Я’ идут снова ‘А’, ‘Б’, ‘В’ и так далее. Таким образом смещение ‘Я’ на 5 даст букву ‘Д’.

На первом этапе для шифрования придумывается первое кодовое слово. В нём считается количество согласных и гласных букв (Ь и Ъ в данной задаче относить к согласным). Все согласные буквы слова, которое надо зашифровать, смещаются на число, равное количеству гласных букв в первом кодовом слове. Все гласные буквы слова, которое надо зашифровать, смещаются на число, равное количеству согласных букв в первом кодовом слове. Например, если кодовое слово — КОТ, то после шифрования слова ГУСЕНИЦА с помощью данного кодового слова получим ДХТЖОКЧВ.

На втором этапе придумывается второе кодовое слово. И получившееся после первого этапа слово (далее называется исходным для второго этапа) кодируется с его помощью. Каждая буква смещается на число, равное позиции определённой буквы (назовём её буквой смещения) в алфавите (алфавит пронумерован с единицы). Буква смещения определяется следующим образом: определяется позиция буквы, которую мы хотим зашифровать в исходном слове (нумерация идёт с единицы); далее это число делится на число букв во втором кодовом слове и берётся остаток от деления. Определяется буква, стоящая во втором кодовом слове на позиции, равной найденному остатку от деления. Эта буква и будет буквой смещения для той буквы, которую мы хотим зашифровать.

Например, с помощью второго кодового слова МАГ полученное после первого этапа шифрования слова ГУСЕНИЦА слово ДХТЖОКЧВ шифруется следующим образом:

1) Определяются буквы смещения для каждой из букв слова ДХТЖОКЧВ. Полученные буквы смещения по порядку: МАГМАГМА. Позиции букв в алфавите: М-14, А-1, Г-4.

2) Зашифрованное слово выглядит так: СЦЦФПОЕГ.

Помогите Андрею зашифровать слово МАРМЕЛАД, если первое кодовое слово - ПОЛИГОН, а второе – ДРУЗЬЯ.

Примечание: Алфавит: АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Ответ: ФХЗШЁОИШ

Решение:

Сперва выполним первый этап шифрования с использованием первого кодового слова ПОЛИГОН. В слове полигон содержится 4 согласные буквы и 3 гласные. Все гласные исходного слова будут смещены на 4 буквы, все согласные – на 3.

М смещается на 3 буквы, получаем П.

А смещается на 4 буквы, получаем Д.

Аналогичным образом смещаем остальные буквы и получаем зашифрованное после первого этапа слово ПДУПИОДЖ.

Далее выполним второй этап шифрования с использованием второго кодового слова – ДРУЗЬЯ. Буквами смещения для каждой из букв зашифрованного на первом этапе слова по порядку будут являться буквы ДРУЗЬЯДР.

Буква Д имеет порядковый номер 5 в алфавите, значит первая буква П будет смещена на пять позиций, получаем Ф.

Буква Р имеет порядковый номер 18 в алфавите, значит вторая буква Д будет смещена на 18 позиций, получаем Х.

Аналогично зашифровываем остальные буквы слова и получаем слово ФХЗШЁОИШ.

7. Информационное моделирование. Теория игр (3 балла) [Магические камни]

Имеется 10 пронумерованных магических камней, разложенных по окружности в следующем порядке (по часовой стрелке):

7, 2, 9, 5, 1, 10, 4, 8, 6, 3

При том между камнями 3 и 7 есть пустое пространство.

Два игрока (Р1 и Р2) ходят по очереди.

Камень считается «доступным», если он непосредственно граничит с пустым местом.

В каждый ход игрок забирает один «доступный» камень и добавляет его значение к своему личному счёту.

Когда камень убирают, на его месте образуется ещё одно пустое место, и соседние камни (если они не были «доступны») могут стать доступными в будущих ходах.

Игра длится, пока все камни не будут взяты. Побеждает игрок с большей итоговой суммой.

Какое максимальное количество очков может получить к концу игры игрок, который может победить независимо от ходов оппонента, если во время игры он (игрок, который победит в этой игре) будет придерживаться своей выигрышной стратегии? Если выигрышных стратегий несколько, игрок выберет ту, при которой наименьшая итоговая сумма, которую он может получить в зависимости от действий оппонента – максимальна среди других стратегий.

Ответ: 32

Решение:

Общая сумма камней равна:

7 + 2 + 9 + 5 + 1 + 10 + 4 + 8 + 6 + 3 = 55

Представим, что мы хотим узнать, сколько баллов может получить игрок, если у него остался какой-то отрезок (последовательность) камней.

Пусть "показатель" — это значение разницы между камнями первого и второго игрока.

Рассмотрим ситуации, когда остался 1 камень, 2 камня, 3 камня и т.д.

Если остаётся один камень (например, только камень с числом 7), то игрок просто его берёт – его "показатель" равен значению камня (7).

Если остаётся два камня (например, 7 и 2), игрок может выбрать либо левый, либо правый. Но надо учесть, что после его хода противник возьмёт оставшийся камень. Поэтому игрок выбирает вариант, при котором разница (его баллы минус баллы противника) будет больше.

В случае камней 7 и 2:

Если мы возьмём камень 2, то его "показатель" будет 2 – 7 = -5, а если камень 7, то 7 – 2 = 5

Таким образом, мы идём «назад»: начинаем с самого простого случая (один камень) и уже зная результаты для 1 и 2 камней, можем определить результат для 3 камней, потом для 4 и так далее.

Построим таблицу 10x10:

	1	2	3	4	5	6	7	8	9	10
1	7	5	4	9	4	6	4	4	2	9
2		2	7	-2	3	7	3	5	5	-2
3			9	4	5	5	-1	11	-3	14
4				5	4	6	10	-2	12	-5
5					1	9	-5	13	-7	10
6						10	6	6	8	9
7							4	4	2	1
8								8	2	5
9									6	3
10										3

Номера столбцов и строк – это номера позиций, на которых лежит камень. В каждой i,j-ой ячейке таблицы записан показатель для соответствующего отрезка с i по j.

Жёлтым цветом обозначены ячейки, которые отвечают за отрезок длины 1, зеленым – отрезки длины 2, бирюзовым – отрезки длины 3 и т.д.

На диагонали выписываем только само значение камня, потому что это случай, когда камень один и мы просто берём его.

Отрезки из двух камней:

Например, для камней 1 и 2 (7 и 2):

Если взять 7, противник получит 2, разница = 7 – 2 = 5

Если взять 2, противник получит 7, разница = 2 – 7 = -5

Выбираем максимум: результат равен 5. Записываем это в таблице для отрезка от 1 до 2.

Отрезки из трёх камней и больше:

Возьмем, например, отрезок с 1 по 3 (7, 2, 9).

Вариант А: игрок берёт 7. Тогда остаётся отрезок (2,9), для которого мы уже вычислили результат (пусть он равен X). Тогда итог будет 7 – X.

Вариант В: игрок берёт 9. Тогда остаётся отрезок (7,2), для которого результат равен 5 (как мы посчитали выше), и итог будет 9 – 5 = 4.

Сравним варианты: если 7-X получается больше 4, выбираем первый вариант; иначе – второй. Записываем итоговую разницу для отрезка (1,3).

Продолжаем так делать для 4, 5, ..., 10 камней в отрезке

Если всё делать аккуратно, в ячейке для всего ряда (от 1 до 10) получим число 9 (разница между первым и вторым игроком)

Вычислим ответ:

Разница между баллами первого и второго игрока равна 9.

Общая сумма камней равна 55.

Получим систему уравнений:

P1 + P2 = 55

P1 - P2 = 9

После решения системы уравнений получим ответ:

P1 = 32

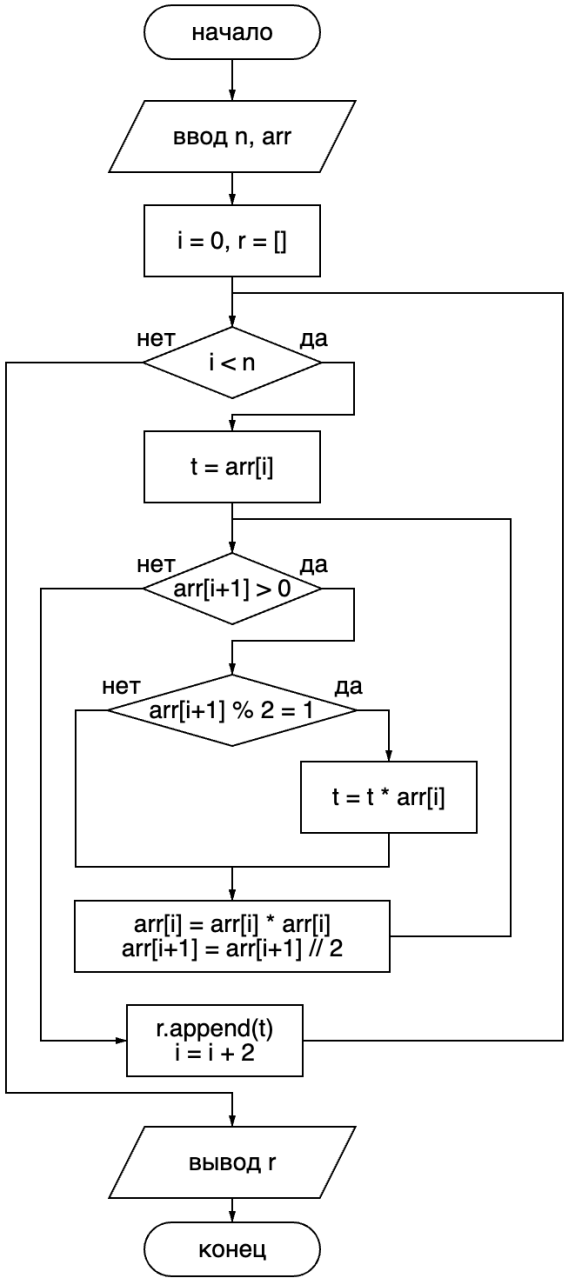
P2 = 23

В ответ было необходимо указать значение max(P1, P2) = 32.

8. Алгоритмизация и программирование. Исполнение алгоритма, заданного в виде блок-схемы (3 балла)

[Матроскин и блок схема]

Коту Матроскину дали блок-схему алгоритма, который получает на вход: число n – число элементов в массиве arr и сам массив arr. Элементы в массиве нумеруются с нуля. Все элементы массива arr – целые положительные числа.



Помогите узнать, какие n и arr Матроскин подал на вход данному алгоритму, если на выходе получил массив r = [4913, 1953125, 177147, 131072, 100000, 40353607]. В ответ запишите без пробелов элементы исходного массива arr по порядку.

Примечание:

a // b – целочисленное деление числа a на число b.

a % b – остаток от деления числа a на число b.

a.append(b) – добавление в конец массива a числа b.

Ответ: 1725831021610478

Решение:

Проанализируем представленный алгоритм. На вход мы получаем какой-то массив. Потом с помощью индекса i начинаем его обход. В переменную t мы кладем наш текущий элемент. Затем мы видим блок, который описывает быстрое возведение в степень, причем в нашем случае, значением степени будет число a[i+1], то есть следующее в массиве за тем, которое мы сейчас рассматриваем. Значение, которое мы будем возводить в степень – a[i], а результат будет как раз лежать в переменной t. Но также заметим, что изначально в переменную t положили значение a[i], следовательно можно сделать вывод, что мы возводим наше число a[i] в степень a[i+1]+1. Далее мы добавление результат возведения в степень в массив и увеличиваем индекс на два. Получается мы последовательно проходимся по парам чисел исходного массива и возводим первое число в степень,

равную второму числу, увеличенному на один. Тогда проанализируем наш выходной массив [4913, 1953125, 177147, 131072, 100000, 40353607]. Либо программно, либо аналитически вычисляем, какой степенью какого числа являются наши значения:

4913 = 17^3, 1953125 = 5^9, 177147=3^{11}, 131072 = 2^{17}, 100000 = 10^5, 40353607= 7^9. Следовательно наш исходный массив: [17, 2, 5, 8, 3, 10, 2, 16, 10, 4, 7, 8]

9. Кодирование информации. Системы счисления (1 балл) [Пиксельное кодирование]

Илья придумал свой собственный способ кодирования слов с помощью трёхцветных пиксельных изображений.

Илья решил, что каждый горизонтальный ряд пикселей изображения будет представлять собой двоичное число. Двоичное число считывается с картинки посимвольно, пиксель за пикселем, по следующему правилу:

Красный (R) пиксель преобразуется в 0,

Зелёный (G) пиксель преобразуется в 1,

Синий (B) пиксель игнорируется (пропускается).

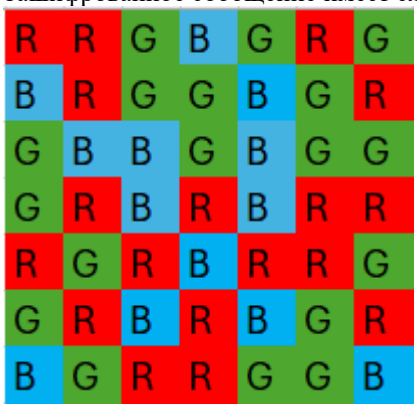
Для раскодирования зашифрованного сообщения необходимо:

1) все двоичные числа, зашифрованные в рядах пикселей, поочередно перевести в десятичную систему счисления;
2) каждое число заменить на остаток от деления этого числа на 33;
3) каждое число интерпретировать как код буквы с помощью таблицы:

13	А	7	К	11	Ч
0	Б	31	Л	6	Ц
9	В	14	М	10	Ч
25	Г	15	Н	24	Ш
2	Д	1	О	30	Щ
16	Е	8	П	22	Ъ
21	Ё	23	Р	27	Ы
29	Ж	5	С	3	Ь
17	З	32	Т	28	Э
18	И	12	У	20	Ю
4	Й	26	Ф	19	Я

После выполнения всех описанных действий должна получиться последовательность из букв, которая представляет собой расшифрованное сообщение.

Зашифрованное сообщение имеет следующий вид:



Помогите Илье расшифровать сообщение.

В ответ запишите полученное сообщение заглавными буквами без пробелов.

Ответ: АМНЕЗИЯ

Решение:

Переведём цветное представление закодированного сообщения в числа в двоичной системе счисления. Запишем каждую строку, начиная сверху. Красный (R) пиксель преобразуется в 0, зелёный (G) пиксель преобразуется в 1, синий (B) пиксель игнорируется (пропускается).

Строка 1: Двоичное представление – 001_101 (знаком “_” обозначается пропуск – пиксель синего цвета). Т. о. получили число 1101_2

Строка 2: _011_10, получается число 1110_2

Строка 3: 1__1_11, получается число 1111_2

Строка 4: 10_0_00, получается число 10000_2

Строка 5: 010_001, получается число 10001_2

Строка 6: 10_0_10, получается число 10010_2

Строка 7: _10011_, получается число 10011_2

Заметим, что все получившиеся двоичные числа идут подряд, каждое следующее больше предыдущего ровно на 1.

Переведём первое двоичное число в десятичную систему счисления: $1101\_2=1*2^3+1*2^2+0*2^1+1*2^0=8+4+0+1=13$

Таким образом, коды букв слова по порядку в десятичной системе счисления: 13, 14, 15, 16, 17, 18, 19.

С помощью таблицы для расшифровки букв по их кодам в десятичной системе счисления найдём соответствующие буквы:

13 – А
14 – М
15 – Н
16 – Е
17 – З
18 – И
19 – Я

Таким образом, расшифрованное сообщение – АМНЕЗИЯ.

10. Сортировка и фильтрация данных (2 балла)
[Сортировка по четности]

Однажды Маше на глаза попался отсортированный по возрастанию массив из 256 чисел: в нем были все числа от 1 включительно до 256 включительно, при чем каждое встречалось ровно 1 раз. Она подумала, что сортировка по возрастанию – это слишком скучно, и решила применить к массиву сортировку, которую придумала сама.

Сортировка Маши работает по следующему алгоритму:

1) Каждая пара соседних элементов сопоставляется друг с другом: в случае, если левый элемент нечетный, а правый четный – элементы меняются местами. Во всех остальных случаях ничего не происходит. Сопоставления происходят последовательно, то есть сначала сопоставление и возможный обмен происходит между первым и вторым элементами, затем – вторым и третьим, далее – третьим и четвертым и так далее до конца массива, итого 255 сопоставлений.

2) Если при выполнении сопоставлений из первого пункта был произведен хотя бы один обмен, то после завершения всех сопоставлений действия из первого пункта алгоритма выполняется снова.

3) Если ни одно из 255 сопоставлений первого пункта не привело к обмену, сортировка считается завершенной.

На какой позиции окажется число 5 после завершения алгоритма? Элементы в массиве нумеруются, начиная с 1.

Ответ: 131

Решение:

Сортировка устойчивая. Это значит, что если в исходной последовательности два числа имеют одинаковую четность, то их порядок в результирующем массиве не будет изменен относительно исходного.

Проанализируем, что происходит в процессе сортировки: первые 128 позиций займут четные числа, при чем в возрастающем порядке – от 2 до 256. А последние 128 позиций займут нечетные числа – так же в возрастающем порядке. Число 1 окажется на позиции 129, число 3 – на позиции 130, а число 5 – на позиции 131.

Отборочный этап. Первый тур (приведен один из вариантов заданий)

1. Теоретические основы информатики (1 балл)
[СУБД]

Выберете из вариантов ниже все названия нереляционных (NoSQL) СУБД:

1) MongoDB
2) Cassandra
3) PostgreSQL
4) CouchDb
5) MariaDB
6) Ни одна из перечисленных

В ответе укажите номера выбранных вариантов ответа через пробел. Пример записи ответа: «1 2».

Ответ: 1 2 4

2. Комбинаторика (2 балла)
[Математический турнир в Цифроландии]

Будем называть особыми числами те, у которых: сумма цифр числа чётная, цифра 1 не стоит на первом месте и число кратно 5

В королевстве Цифроландия король устроил математический турнир для своих подданных. Он объявил, что тот, кто сможет составить наибольшее количество особых четырёхзначных чисел из заданных цифр, получит титул Главного Математика Королевства.

Пусть заданы цифры: 1, 2, 3, 4, 5, 6, 7 и каждую цифру можно использовать не более одного раза в числе.

Сколько четырёхзначных особых чисел существует? В ответ укажите количество таких чисел

Ответ: 52

3. Комбинаторика (2 балла)
[Подбор токена]

Банк разрабатывает новую систему токенов безопасности. Токен представляет собой последовательность из 15 элементов, записанных через дефис. Элементы нумеруются с 1. На чётных позициях находится одна из английских букв: A, B, C, D, E, F, G, H, I, K, буквы могут повторяться. На нечётных позициях находятся числа из диапазона от 0 и до 9 включительно, числа в одном токене повторяться не могут и расположены в таком порядке, что при движении слева направо

по элементам токена каждое число в токене меньше предыдущего числа (кроме числа на позиции 1, потому что у него нет предыдущего числа).

Пример верного токена:

9-K-8-C-6-D-5-K-4-F-2-G-1-C-0

Известно, что злоумышленник пробует подобрать токен методом перебора. В день он может проверять не больше 10000 токенов. Банк хочет узнать, какое максимальное число дней может потребоваться злоумышленнику, чтобы методом перебора подобрать нужный токен.

Злоумышленник перебирает только возможные токены, условия для которых соблюдаются. То есть он не будет, например, перебирать токены у которых на чётных позициях цифры.

В ответ напишите максимальное число дней, которое может потребоваться злоумышленнику на подбор пароля.

Ответ: 45000

4. Основы логики (3 балла)

[Пары функций]

Известно, что:

$$\big(\text{НЕ}(\boldsymbol{a})\ \text{ИЛИ НЕ}(\boldsymbol{b})\ \text{ИЛИ НЕ}(\boldsymbol{c})\ \text{ИЛИ}\ \boldsymbol{A}(\boldsymbol{d},\boldsymbol{e},\boldsymbol{f})\big)\ \text{И}\ \big(\text{НЕ}(\boldsymbol{d})\ \text{ИЛИ НЕ}(\boldsymbol{e})\ \text{ИЛИ НЕ}(\boldsymbol{f})\ \text{ИЛИ}\ \boldsymbol{B}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\big) = \mathbf{1},$$

где A(d, e, f) и B(a, b, c) – логические функции от трех переменных, связанные между собой следующим условием:

$$\boldsymbol{A}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ \text{И}\ \boldsymbol{B}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \mathbf{0}$$

Сколько существует пар функций A и B, чтобы выполнялось указанное тождество? Пара – упорядоченный набор из двух функций, функции в паре могут повторяться. В ответ запишите количество таких пар или 0, если их не существует.

Ответ: 0

5. Кодирование. Шифрование (3 балла)

[Битовый шифр]

Петя придумал следующий шифр:

1. Каждой букве ставится в соответствие десятичное число (кодовый номер)
2. Перевести номера соответствующих букв в двоичный код (в двоичную систему счисления).
3. Чтобы закодировать последовательность из двух букв необходимо произвести следующий алгоритм:
   1. Взять два соответствующих двоичных кода (последовательности битов) A и B соответственно.
   2. Завести переменную X, которая изначально равна 0.
   3. Для каждого бита второго числа (B), начиная с самого последнего бита и двигаясь справа налево, выполнять следующие шаги:
      3.1 Если бит равен 1, добавить первое число (A) к переменной X в двоичной системе счисления
      3.2 Добавить к первому числу (A) один ноль в конец.
   4. Результатом кодировки является число, записанное в переменную X в двоичной системе счисления.

Пример: A = 13, B = 5. Тогда последовательность AB будет шифроваться следующим образом:

1. $A = 13_{10}=1101_2$, $B=5_{10}=101_2$
2. X=0
3. Последний бит B = 1 => X=1101, A=11010
4. Следующий бит B = 0 => X=1101, A=110100
5. Следующий бит B = 1 => X=1101+110100=1000001, A=1101000

Для кодирования слова, в котором более двух букв, сначала первые две буквы обрабатываются по указанному алгоритму, а потом последовательно результат, полученный на предыдущем шаге, подвергается обработке алгоритма с двоичным кодом последующей буквы. Тогда последний результат работы алгоритма и будет являться шифром для слова. Так же Петя гарантирует, что результат работы алгоритма двух букв не может являться кодовым номером для какой-либо буквы.

Петя зашифровал слово из четырех букв и сказал Васе, что оно начинается с буквы Г и имеет шифр 120. Помогите Васе понять, какое количество разных расшифровок сообщения существует? Различными считаются такие сообщения, которые различаются хотя бы одной буквой. Есть таблица соответствий букв и их кодовых номеров:

А	Г	Р	М	К	И	У
2	3	4	5	6	7	8

Использовать в сообщении буквы не из этой таблицы запрещено.

Ответ: 4

6. Системы счисления (1 балл)

[Система счисления с основанием n]

Петя записал число в системе счисления с основанием n, и получил число 121. Затем он перевёл это же число в систему счисления с основанием n − 1 и получил число 210.

Найдите основание n.

Ответ: 5

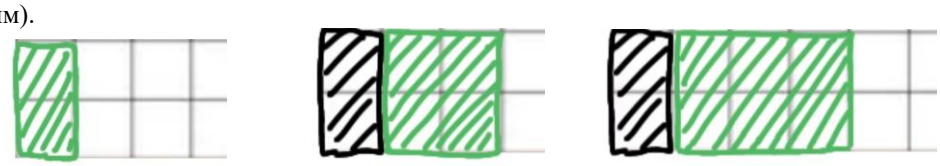
7. Теория игр. Моделирование (2 балла)

[Игра на клетчатом поле]

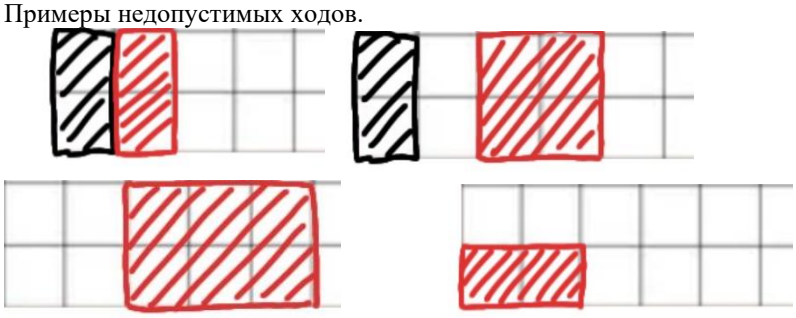
Прохор, Василиса и Тоня придумали следующую игру. Есть клетчатое поле 372*2. Игроки ходят по очереди. Первым ходит Прохор, второй – Василиса, третьей – Тоня. Игрок в свой ход может закрасить на этом поле квадратик 2*2 клеточки, прямоугольник 1*2 клеточки (только вертикальный) или прямоугольник 2*3 (только горизонтальный). При этом нельзя

закрашивать точно такую же фигуру, как закрасил игрок до этого, а выбранная для раскрашивания фигура должна своей левой стороной примыкать либо к границам поля, либо к уже закрашенным фигурам.

Примеры допустимых ходов (черная фигура обозначает фигуру, закрашенную предыдущим игроком, цветная – текущим).



Примеры недопустимых ходов.



Побеждает тот, кто закрасит последний квадратик на поле.

Известно, что никто из игроков не поддается. Кто победит при правильной игре и какое максимальное количество его ходов ему для этого потребуется? Запишите в ответ два числа через пробел – номер игрока, который победит (1 – Прохор, 2 – Василиса, 3 – Тоня) и количество ходов. Обратите внимание, что требуется написать количество ходов этого конкретного игрока до его победы, а не общее).

Пример ответа: 1 2

Ответ: 3 62

8. Кодирование информации. Системы счисления (2 балла)

[Картина в троичной системе счисления]

Вариант 1

Алиса любит работать с картинками. Картинки хранятся в памяти компьютера по пикселям, каждый пиксель характеризуется тремя числами - количество входящего в состав этого пикселя красного, зеленого и синего цвета соответственно. Каждое из этих чисел имеет значение от 0 до 255 и обозначается соответствующей цвету буквой r (red), g (green), b (blue).

Алиса придумала свой способ шифрования двоичных сообщений с помощью таких картинок: в трёх числах цветов каждого пикселя, зашифрована одна цифра равная либо "1", либо "0" по следующему правилу: если у значений двух и только двух из трёх цветов пикселя при представлении их в троичной системе счисления, два младших разряда равны 0 (например, число 288 в троичной системе счисления равно 101200, два младших разряда числа равны 0), то в пикселе зашифрована 1. Если пиксель не соответствует этому условию, то в пикселе зашифрован 0.

В ответ запишите расшифрованную последовательность из нулей и единиц в порядке соответствующих пикселей в изображении, построчно, слева направо, сверху вниз. (Пример ответа: 111100011)

R = 254 G = 126 B = 32	R = 225 G = 221 B = 221	R = 224 G = 108 B = 72
R = 81 G = 234 B = 36	R = 153 G = 207 B = 102	R = 216 G = 126 B = 32
R = 244 G = 189 B = 108	R = 202 G = 126 B = 44	R = 100 G = 45 B = 234

Ответ: 001011101

9. Кодирование информации. Звук (1 балл)

[Спор об mp3 файлах]

Соня и Кристина узнали, что в mp3 файлах помимо самой закодированной музыки также хранятся текстовые поля – так называемые теги. Они расположены в начале файла и имеют строго определенный информационный объем независимо от своего содержания:

Название композиции: 400 бит

Имя исполнителя: 336 бит

Название музыкального альбома: 248 бит

Соня, когда узнала это, возмутилась: «Зачем хранить столько информации внутри файла любимой песни, если всю эту информацию ты и так знаешь? Это же такая потеря места на компьютере!»

На это Кристина возразила, сказав, что если уж экономить место, то было бы гораздо практичнее хранить аудио не двухканальным, а одноканальным.

Помогите рассудить девочек: на примере песни длиною 90 секунд, в файле которой содержится информация о названии композиции, исполнителе и альбоме, изначально записанной в стерео режиме при частоте 40 кГц и глубине кодирования 16 бит, определите, что больше уменьшит объем хранимой информации и насколько: уменьшение числа каналов до одного или удаление из файла трех текстовых полей, описанных выше?

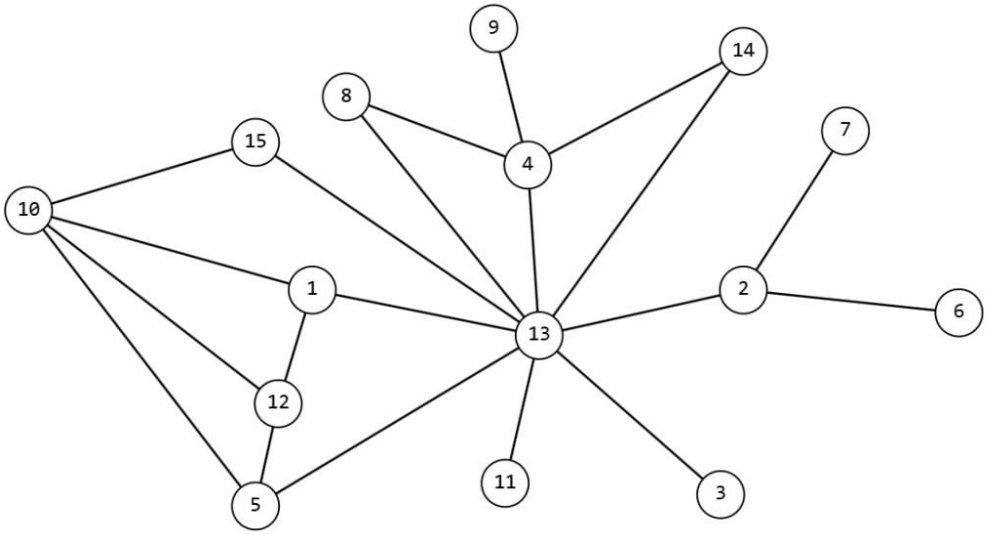
В ответ запишите разницу между максимальным и минимальным из объемов двух вариантов модификации музыкального файла. Ответ дайте в битах.

Ответ: 57599016

10. Моделирование. Графы (1 балл)
[Камеры на перекрестках]

Вариант 1

В городе есть перекрёстки, соединённые дорогами. Перекрёсток – это любая вершина графа, в том числе та, где сходится только две дороги. Перекрёстки пронумерованы от 1 до 15. Дороги между ними следующие:



Городская администрация хочет установить камеры наблюдения на минимальном количестве перекрёстков так, чтобы каждая дорога была под наблюдением (то есть хотя бы один из её концов имел камеру). Камеры у нас технологичные, поэтому снимают перекрёсток на 360 градусов (т.е. камера, находящаяся на перекрестке, одновременно снимает все дороги, которые сходятся к этому перекрёстку).

На каких перекрёстках нужно установить камеры?

Формат ответа: перекрёстки записываются в порядке возрастания слитно

Пример ответа: 1234

Ответ: 24101213

Отборочный этап. Второй тур (приведен один из вариантов заданий)

1. Архитектура компьютера (1 балл)
[Архитектура процессоров]

Используя поиск информации в глобальной сети Интернет, определите, какой из следующих типов архитектур процессоров предполагает выполнение одной инструкции за один такт.

1) CISC (Complex Instruction Set Computing)
2) RISC (Reduced Instruction Set Computing)
3) VLIW (Very Long Instruction Word)
4) MISC (Minimal Instruction Set Computing)

В ответе укажите одно целое число – номер верного утверждения.

Ответ: 2

2. Архитектура компьютера (2 балла)
[Паттерны программирования]

Используя поиск информации в глобальной сети Интернет, определите, какие паттерны НЕ относятся к структурным паттернам проектирования программного обеспечения.

1. Адаптер (Adapter)
2. Стратегия (Strategy)
3. Декоратор (Decorator)
4. Наблюдатель (Observer)
5. Фасад (Facade)

В ответ запишите все выбранные варианты ответа в порядке возрастания без пробелов и знаков препинания, например "1245"

Ответ: 24

3. Моделирование (1 балл)
[Магический отдел]

В обычной на первый взгляд библиотеке Санкт-Петербурга есть отдел с магическими книгами, они расположены на 10 полках по 7 книг на каждой.

Книги пронумерованы как единая для всех книг последовательность подряд идущих чисел, начиная с 1, например, если бы у нас было по 2 книги на каждой полке и всего полок 3, то расположение книг будет выглядеть следующим образом (|1 2|3 4|5 6|). Волшебнице Даше больше всего нравятся книги с номерами, кратными 5.

В начале каждой недели Даша приходит в библиотеку и забирает к себе домой книги по следующему алгоритму:

Она выбирает полку, забирает все книги с неё, а также по 1 книге со всех других полок, таким образом, чтобы взять как можно больше книг с ее любимыми номерами.

В конце недели, ровно через 7 дней, Даша возвращает все книги и осматривает все полки отдела.

Как только она обнаружит, что непрочитанные книги с номерами, кратными 5, закончились, она перестанет ходить в эту библиотеку.

Через какое минимальное количество дней от первого посещения Даша примет решение перестать ходить в библиотеку?

В ответе необходимо указать число, кратное 7.

Ответ: 14

4. Моделирование (2 балла)
[Трудности перевода]

Костя создал робота, запоминающего слова, и решил обучить его английскому языку. Для этого он составил список из 100 000 слов, которым он хотел научить робота. Слова в списке пронумерованы, начиная с 1.

Обучение проходит следующим образом: Костя показывает роботу новое слово, робот его запоминает, а затем Костя проверяет пройденный ранее материал: для этого он поочередно называет все изученные ранее слова, а робот – называет их переводы. Если робот назвал перевод неправильно, то считается, что робот забыл это слово, в таком случае Костя ставит в своем списке рядом с этим словом пометку, чтобы больше это слово не спрашивать. Порядковые номера остальных слов от этого действия не изменяются. Изучать заново это слово робот не будет.

Первые 256 слов всё шло хорошо: робот запоминал их все и каждый раз безошибочно отвечал все пройденные до этого слова.

Однако дальше начались проблемы. Оказалось, что:

- после того, как робот учит слово с порядковым номером, делящимся на 5 без остатка, он забывает все слова под номерами, также делящимися на 5 без остатка из изученных ранее;
- после того, как робот учит слово с порядковым номером, делящимся на 3 без остатка, он забывает все слова под номерами, также делящимися на 3 без остатка из изученных ранее;

Назовем особым условием ситуацию, когда порядковый номер изучаемого слова не делится без остатка ни на 3, ни на 5. Если порядковый номер изучаемого слова удовлетворяет особому условию, тогда робот забывает первые 2^n слов из тех слов списка, которые он еще помнит. Здесь n — это порядковый номер такой ситуации: например, для слова №257 - первое слово после 256-го, которое удовлетворяет особому условию, n=1, тогда робот должен будет забыть первые 2^1 еще не забытых слов из списка слов. Если слов, которые робот может забыть, меньше, чем 2^n, то робот забывает их все.

Примечание: если робот учит слово с номером, делящимся и на 3, и на 5, то он забывает все слова из изученных ранее, номера которых делятся на 3 и/или на 5.

Робот не забывает слова мгновенно: независимо от номера слова, которое он сейчас учит, и того, насколько этот номер соответствует тем или иным условиям, изучаемое сейчас слово не будет забыто как минимум до тех пор, пока не будет выучено следующее слово.

Несмотря на все выявленные трудности, Костя решил продолжить обучать робота до тех пор, пока в его памяти не останется ровно 1 слово (иными словами, пока не наступит ситуация, когда после изучения очередного слова робот не забудет все остававшиеся в памяти слова до ново изученного) или пока не закончатся слова в его списке.

Какой порядковый номер у слова, которое робот изучит последним? В ответ запишите одно число — порядковый номер этого слова.

Ответ: 269

5. Моделирование (3 балла)
[Конечные состояния]

Программе на вход поступает одно число, она проводит его анализ по приведённому ниже алгоритму и приходит к одному из двух конечных состояний (состояние 1 или состояние 2).

Если сумма цифр на нечётных позициях нечётная, то программа приходит в первое конечное состояние и завершает работу. Цифры в числе нумеруются с нуля слева направо.

Если сумма цифр на нечётных позициях чётная, то проверяется сумма цифр на чётных позициях и если она чётная или в числе меньше пяти цифр, то программа приходит во второе конечное состояние и завершает работу. Иначе из числа удаляется цифра под номером n // 2, где n – количество цифр в числе, а // — деление с округлением вниз до целого числа. Например, из числа 12345 получится число 1245. После чего программа начинает выполнять свою работу с самого начала — на этот раз для полученного нового числа.

Программе передаются числа от 10000 включительно и до 20100 не включительно. В ответ запишите, сколько раз программа завершится в первом конечном состоянии.

Ответ: 6325

6. Моделирование (3 балла)
[Матроскин на поле]

Кот Матроскин прыгает по клеткам поля. Поле окружено стенами со всех сторон. В каждой клетке поля лежит какое-то количество денег, равное числу, записанному в клетке. Для перемещения по полю действуют следующие правила:

- Из каждой клетки кот берет себе все деньги (т.е. после того, как он уходит с клетки, в ней 0 денег).
- Если сумма денег на лапах у кота (вместе с деньгами с той клетки, на которой он сейчас стоит) при делении на 4 дает остаток 0 – перемещаемся на клетку вправо; 1 – влево; 2 – вниз; 3 – вверх.
- Если кот врезается в стену – он останавливается и не прыгает дальше.
- Кот может стартовать с любой клетки, кроме правой нижней.

46	13	32	36	31	25	24	38	20	29
44	45	13	40	3	17	1	31	39	16
3	45	17	30	31	40	10	5	20	34
30	16	7	6	42	10	5	29	45	15
40	36	11	5	36	37	44	28	46	22
37	23	12	24	33	3	12	37	22	36
8	13	46	3	33	18	24	22	48	40
46	2	29	4	27	3	38	35	6	20
39	33	14	25	30	41	49	44	17	17
25	49	17	47	8	1	12	38	32	27

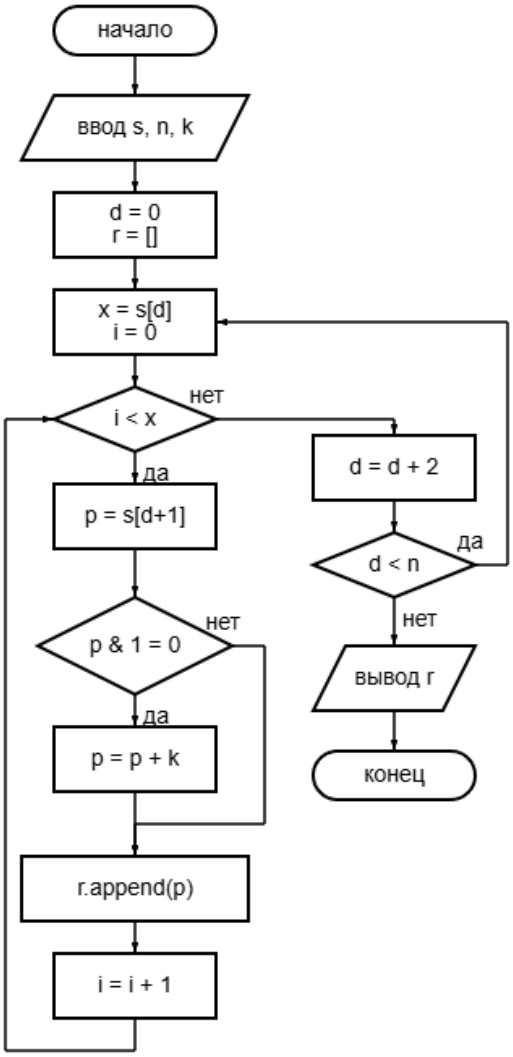
Сколько существует стартовых позиций, из которых кот сможет дойти по правой нижней клетки?

Ответ: 3

7. Алгоритмизация и программирование (3 балла)
[Зажатый массив]

На вход программе передаётся массив s, состоящий из n чисел, и два числа n и k, а на выходе программа возвращает массив r, также состоящий из чисел.

Проанализируйте работу алгоритма и скажите, какой массив s был передан на вход, если известно, что n = 12, k = 4, r = [6, 6, 6, 4, 5, 11, 11, 11, 11, 11, 12, 12, 12, 12, 8].



Примечание:

1. “p & i”– операция побитого И между числами p и i. Например 3 & 2 = 2.
2. “r = []” – инициализация пустого массива.
3. “r.append(i)” – добавление в конец массива элемента i.
4. “s[i]” – обращение к элементу массива под номером i.

В ответ запишите числа массива s подряд без пробелов и знаков препинания. Пример ответа: 123456.

Ответ: 3210155114814

8. Алгоритмизация и программирование. (1 балл)

[Тайны ИТМО]

Древнее подземелье ИТМО состоит из зачарованных залов, которые обозначены как R0, R1, R2, R3, R4, R5, R6, R7. Цель поступающих — попасть из зала R0 в зал R4, где сохранены древние знания, как решать любую задачу на олимпиаде.

Поступающие знают два заклинания, назовём их a, b. При произнесении очередного заклинания в зачарованном зале могут открыться некоторые ворота из этого зала. Поступающие иногда могут пройти сразу в несколько ворот (будем считать, что они разделились, чтобы проверить все ходы параллельно).

Список переходов в зависимости от зала:
R0:
При прочтении заклинания:
a — открывает ворота в R1 и в R2;
b — открывает ворота в R6.

R1:
При прочтении заклинания:
a — открывает ворота в R6;
b — открывает ворота в R2.

R2:

При прочтении заклинания:
a — открывает ворота в R6;
b — открывает ворота в R3.

R3:
При прочтении заклинания:
a — открывает ворота в R7;
b — открывает ворота в R6.

R4 — наша цель, из неё переходов нет.

R5:
При прочтении заклинания:
a — открывает ворота в R4;
b — открывает ворота в R6.

R6:
При прочтении заклинания:
a — открывает ворота в R6;
b — открывает ворота в R6.
(по сути, это тупик, вы застряли)

R7:
При прочтении заклинания:
a — открывает ворота в R6;
b — открывает ворота в R5.

Вам предстоит определить кратчайшую последовательность из заклинаний, которая приведёт из зала R0 в зал R4.
Ответ: ababa

9. Сортировка и фильтрация (3 балла)
[Матроскин и таблица]

Кот Матроскин составил какую-то последовательность чисел и записал ее в электронной таблице в ячейках A1:A15.

Затем в ячейках B1:B14 он записал суммы значений ячеек из столбца A, находящихся в той же строке и на строку ниже. Например, в ячейке B1 лежит сумма значений A1 и A2. В ячейках C1:C13 он записал суммы значений ячеек из столбца B, находящихся в той же строке и на строку ниже. Например, в ячейке C1 лежит сумма значений B1 и B2.

Потом пришел почтальон Печкин и решил испортить таблицу Матроскина. Он отсортировал столбцы A и B по возрастанию независимо друг от друга (то есть, при сортировке столбца, значения других столбцов никак не менялись).

На картинке ниже приведен фрагмент таблицы Матроскина после вмешательства Печкина. Помогите Матроскину восстановить исходную последовательность. В ответ запишите сумму значений, которые изначально находились в ячейках A1 и A15.

	A	B	C
1	-17711	-6765	13
2	-6765	-2584	-21
3	-2584	-987	34
4	-987	-377	-55
5	-377	-144	89
6	-144	-55	-144
7	-55	-21	233
8	34	34	-377
9	89	89	610
10	233	233	-987
11	610	610	1597
12	1597	1597	-2584
13	4181	4181	4181
14	10946	10946	
15	28657		

Ответ: 28691

10. Сортировка и фильтрация (2 балла)
[Импорт]

Компания "CLatghotiam" анализирует данные о продажах автомобилей за первую половину календарного года в шести регионах. Данные представлены в виде таблицы. Регионы пронумерованы с 1 по 6 сверху вниз, а месяцы с 1 по 6 слева направо.

Каждое число в таблице представляет количество проданных автомобилей в соответствующем регионе в соответствующий месяц.

12	7	9	14	8	10
15	6	11	13	9	12
10	8	7	16	5	11
14	9	10	12	7	13
13	5	12	11	10	9
9	10	8	15	6	14

Для оценки эффективности проделанной работы вычисляется число K.

Число K вычисляется как среднее арифметическое значение продаж определенных регионов в определенные месяцы из рассматриваемых данных.

Чтобы понять, нужно ли учитывать продажу n-го региона в m-ый месяц, необходимо, чтобы она удовлетворяла следующим условиям:

1) Среднее значение продаж n-го региона больше 10,7 (n — номер региона).

2) Если рассмотреть продажи всех регионов за m-ый месяц, то не более n регионов в этом месяце получили более высокий показатель продаж (m — номер месяца).

Вычислите число K, учитывая только те продажи, которые удовлетворяют указанным условиям. Округлите результат до одного знака после запятой (с математическим округлением).

Пример ответа: 1023.3

Ответ: 11.2