Открытая олимпиада школьников (информатика)
(№67 Перечня олимпиад школьников, 2023/2024 уч.год)

Содержание

Задания для 11 класса
Заключительный этап (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (3 балла)
[Где-то далеко…]

Петя решил сделать свой вклад в онлайн-энциклопедию целочисленных последовательностей. Последовательность, которую решил сгенерировать Петя, устроена следующим образом:

Первый член последовательности, это число 73_{19}, где 19 – основание системы счисления. Для получения каждого следующего члена последовательности Петя увеличивает на 1 каждую цифру и увеличивает на 1 основание системы счисления. Например, второй член последовательности будет записан как 84_{20}. Определите, чему равна сумма всех членов последовательности вплоть до члена с номером $(10^{10}+1)_{10}$ включительно. Посчитайте и укажите в ответе сумму цифр в десятичной записи этого числа.

Ответ: 81

Решение:

Рассмотрим (n+1)-ый член последовательности: ab_c. Тогда исходя из условия a=(7+n), b=(3+n), c=(19+n). Используя развернутую форму записи числа, получим: (7+n)*(19+n)+(3+n) = 133 + 19*n + 7*n + n^2 + 3 + n = n^2 + n*27 + 136.

Тогда искомая сумма первых $(10^{10}+1)$ членов последовательности будет равна $\sum_{i=0}^{10^{10}} i^2 + 27 * \sum_{i=0}^{10^{10}} i + 136 * (10^{10} + 1)$

С вычислением $\sum_{i=0}^{10^{10}} i$ не возникнет сложностей, это просто сумма первых 10^{10} натуральных чисел и равна $\frac{10^{10}*(10^{10}+1)}{2}$. Осталось посчитать первую сумму. Можно знать формулу или вывести её и доказать по индукции: $1^2 + 2^2 + \cdots + n^2 = \frac{n*(n+1)*(2*n+1)}{6}$. Тогда в нашем случае получится: $\sum_{i=0}^{10^{10}} i^2 = \frac{10^{10}*(10^{10}+1)*(2*10^{10}+1)}{6}$.

Осталось вычислить значение и посчитать сумму цифр. Поскольку речь идет о больших натуральных числах, удобнее это сделать на Python, хотя можно и применить умение пользоваться «длинной арифметикой» на других языках. На Python для вычисления можно использовать, например, такой код:

```
R = 10**10*(10**10+1)*(2*10**10+1))//6+27*(10**10*(10**10+1))//2+136*(10**10+1)
print(R)
ans = 0
while R>0:
    ans += R % 10
    R //= 10
print(ans)
```

Обратим внимание, что используются операторы целочисленного деления. Это сделано потому, что в случае вещественного деления за счет ограничений на хранение вещественного числа получится ошибка в вычислениях, а при этом легко доказать, что обе операции деления всегда будут давать целочисленный результат, а значит можно использовать целочисленное деление.

В результате выполнения кода получим ответ 81.

2. Кодирование информации. Объем информации (1 балл)
[Цифровой диктофон]

Друг Пети, Павел, пытается сконструировать цифровой диктофон и просит Петю написать прошивку для кодирования и сохранения в памяти оцифрованного аудиосигнала. Петя решил, что будет записывать данные без сжатия и оцифровывать аудиосигнал с частотой дискретизации 88200 Hz, выбрав такую глубину кодирования, чтобы в каждый отсчет времени сохранялось одно из возможных 65536 значений сигнала (для записи значения сигнала в каждый отсчет времени Петя использует минимальное, одинаковое для всех возможных значений количество бит). Поскольку Петя предполагает использовать пару микрофонов, Павел решил записывать звук двухканальным, сохраняя оцифрованный аудиосигнал с указанными параметрами независимо для каждого канала. Опытный Вася обратил внимание Пети на две возможности для уменьшения памяти. Во-первых, можно уменьшить частоту дискретизации в два раза, а во-вторых, записывать с выбранной глубиной кодирования только один канал, а для второго канала записывать для каждого отсчета времени только разность значения сигнала со значением в первом канале, считая, что для этого хватит 256 возможных значений (для записи разности сигналов в каждый отсчет времени предлагается также использовать минимальное, одинаковое для всех возможных значений разности количество бит). Петя принял оба предложения Васи и обнаружил, что для аудиосигнала длительностью t секунд удалось сэкономить больше 20 МБайт памяти. Определите минимальное **целое** значение t, при котором это возможно. В ответе укажите целое число.

Примечания:

1. При записи оцифрованного сигнала в память не записывается никакая дополнительная информация.
2. 1 МБайт=2^{20} байт.

Ответ: 96

Решение:

Запишем формулу для определения информационного объема в первоначальном формате записи:

2 * t * 88200 * $\log_2(65536)$ = t * 88200 * 2 * 16 бит.

После изменения формата частота дискретизации составит 88200/2=44100 Hz, а глубина кодирования для второго канала станет равна $\log_2(256)$ = 8 бит. Следовательно, формула для определения информационного объема будет: t * 44100 * 16 + t * 44100 * 8 = t * 44100 * 24 бит.

Определим значение t, при котором экономия памяти составила бы ровно 20 МБайт:

t * 88200 * 2 * 16 - t * 44100 * 24 = 12 * 1024 * 1024 * 8

t = 20 * 1024 * 1024 * 8 / (88200 * 2 * 16 – 44100 * 24) = 95,11 секунд

Следовательно, если бы длительность аудиосигнала была 95 секунд, разность составила бы меньше 20 МБайт, значит ответ – 96 секунд.

3. Основы логики (2 балла) [ABCD…XYZ]

Определены четыре логические функции:

$A(X,Y,Z) = X \wedge Y \rightarrow \bar{Y} \wedge Z$

$B(X,Y,Z) = Y \wedge Z \rightarrow \bar{Z} \wedge X$

$C(X,Y,Z) = \bar{Z} \wedge Y \rightarrow \bar{Y} \wedge X$

$D(X,Y,Z) = \bar{Z} \wedge X \rightarrow \bar{X} \wedge Y$

Сколько существует неэквивалентных друг другу логических функций $F(A,B,C,D)$, таких, что $F\big(A(X,Y,Z),B(X,Y,Z),C(X,Y,Z),D(X,Y,Z)\big) = Y \rightarrow X \wedge Z$?

В ответе укажите целое неотрицательное число (если подходящей функции F не существует, в ответе укажите 0).

Ответ: 1024

Решение:

Построим таблицы истинности определенных в условии функций $A(X,Y,Z)$, $B(X,Y,Z)$, $C(X,Y,Z)$ и $D(X,Y,Z)$:

X	Y	Z	A(X,Y,Z)	B(X,Y,Z)	C(X,Y,Z)	D(X,Y,Z)
0	0	0	1	1	1	1
0	0	1	1	1	1	1
0	1	0	1	1	0	1
0	1	1	1	0	1	1
1	0	0	1	1	1	0
1	0	1	1	1	1	1
1	1	0	0	1	0	0
1	1	1	0	0	1	1

Дополним таблицу столбцом $F(X,Y,Z) = Y \rightarrow X \wedge Z$

X	Y	Z	A(X,Y,Z)	B(X,Y,Z)	C(X,Y,Z)	D(X,Y,Z)	F(X,Y,Z)
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	1	1	0	1	0
0	1	1	1	0	1	1	0
1	0	0	1	1	1	0	1
1	0	1	1	1	1	1	1
1	1	0	0	1	0	0	0
1	1	1	0	0	1	1	1

Сразу обратим внимание, что получилось несколько строк, в которых значения A, B, C и D одинаковые (в случае этого варианта задачи – три строки, в которых все четыре функции принимают истинные значения). Важно отметить, что во всех строках функция F принимает одинаковое значение. Если бы это было не так, это означало бы, что подходящей функции $F(A,B,C,D)$ не существует и ответ был бы 0.

Заполним те строки таблицы истинности функции $F(A,B,C,D)$, которые нам известны из предыдущей таблицы:

A	B	C	D	
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	1
0	1	0	0	0
0	1	0	1	
0	1	1	0	
0	1	1	1	

1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	0
1	1	0	0	
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Обратим внимание, что для 10 комбинаций значений A, B, C и D функция F исходя из значений X, Y и Z не определена. Следовательно, может принимать любое из возможных двух значений (истина или ложь) для каждой такой строки. Значит таких функций существует 2^{10}=1024.

4. Кодирование информации. Формальные исполнители (1 балл) [6 букв]

Есть исходная строка S, состоящая из 6 символов. Результирующая строка R изначально пустая и формируется следующим образом: N раз выполняются следующие два шага:

1. Дописать в конец строки R строку S.
2. Циклически сдвинуть строку S на один символ влево.

Например, для строки S=’abcdef’ и N=4 получится строка R=’abcdefbcdefacdefabdefabc’.

Для некоторой строки S получили результирующую строку R для N=10^{10}.

Известны некоторые символы в строке R (нумерация символов строки начинается с 1 слева направо):

Номер символа	Значение
10^8+2	a
10^8+4	c
10^8+8	b
10^8+16	d
10^8+3	f
10^8+5	e

Определите и введите в ответ строку S, для которой это возможно.

Если вариантов таких строк несколько, введите любую подходящую.

Если такой строки не существует, введите в ответ NULL.

Ответ: cedabf

Решение:

Обратим внимание, что, поскольку в исходной строке 6 символов, возможно сделать последовательно 6 циклических сдвигов, после чего опять получится исходная строка. Следовательно, в результирующей строке будут повторяться одинаковые последовательности из 36 символов.

Построим такую последовательность (вручную или с помощью простой программы), взяв за основу строку «123456»:

123456234561345612456123561234612345

Возьмем первый номер из таблицы: 10^8+2 и поделим на 36 с остатком. Остаток будет равен 30. Следовательно, это 30-ый символ в этой строке – символ «4». Следовательно, четвертый символ в исходной строке, символ “a”.

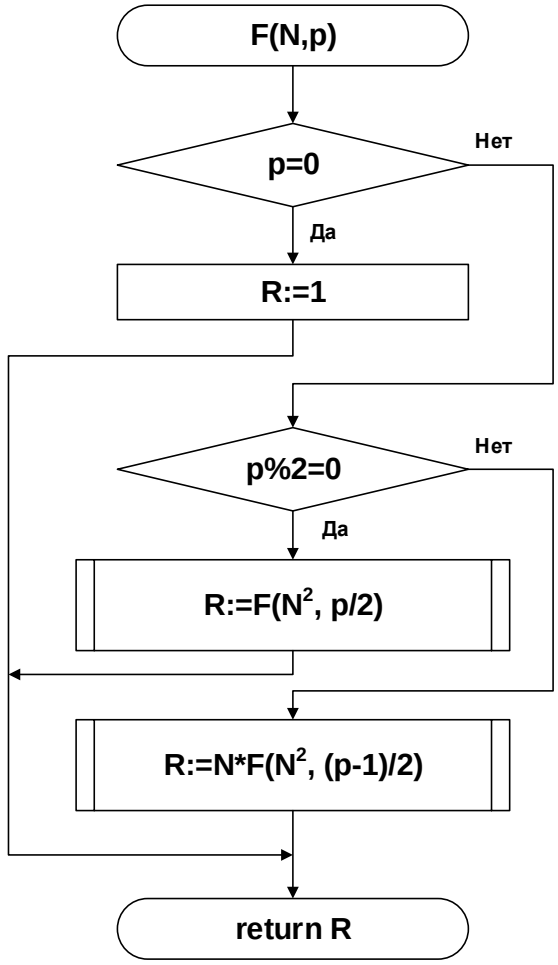
Аналогично поступим с остальными номерами символов и получим следующую таблицу:

Номер символа в результирующей строке	Значение	Номер символа в исходной строке
10^8+2	a	4
10^8+4	c	1
10^8+8	b	5
10^8+16	d	3
10^8+3	f	6
10^8+5	e	2

Обратим внимание, что мы определили все 6 символов исходной строки. Запишем их в правильном порядке и получим ответ: cedabf.

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (2 балла) [Рекурсия]

Дана блоксхема алгоритма, реализованного в виде рекурсивно вызываемой функции:



Найдите такую пару целых положительных чисел N и p (известно, что p>1), чтобы вызов F(N,p) вернул число 387420489. Если таких пар существует несколько, найдите ту, у которой максимальное значение N. В ответе укажите через пробел сначала значение N и затем значение p. Если такой пары не существует, укажите в ответе NULL.

Ответ: 19683 2

Решение:

Проанализировав алгоритм, представленный в виде блок-схемы, можно понять, что он реализует возведение числа N в степень p. Следовательно, нужно подобрать такие N и p, чтобы $N^p = 387420489$.

Для этого разложим данное в условие число на сомножители, например, с помощью такого программного решения:

```
ans=[]
d=2
while Z>1:
    if Z%d==0:
        ans.append(d)
        Z//=d
    else:
        d+=1
print(ans)
```

Результатом будет список: [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3], то есть 18 чисел 3. Значит $387420489 = 3^{18}$.

Но в условии сказано, что необходимо найти пару с максимальным значением N.

Заметим, что 18 = 9*2. Тогда $3^{18} = (3^9)^2 = 19\,683^2$. И это пара с максимальным значением N, поскольку значение p – минимально возможное (по условию p>1). Следовательно, ответ: 19683 2.

6. Телекоммуникационные технологии (2 балла). [Обрыв канала]

В ОС семейства GNU/Linux существует возможность объединения сетевых интерфейсов (сетевых карт) в группы (*bonding*). Ее используют для балансировки нагрузки при передаче и обеспечения отказоустойчивости.

В результате объединения в системе создается виртуальный сетевой интерфейс `bond`. С точки зрения остальных слоев сетевого стека этот интерфейс – обычная сетевая карта, однако при передаче данных через нее реальная передача данных осуществляется через тот или иной физический сетевой интерфейс по выбору ядра операционной системы. Выбор физического интерфейса зависит от режима *bonding*.

Существует 7 режимов *bonding*. Один из них – `balance-xor`. Когда физические интерфейсы объединяются в этом режиме, для выбора физического интерфейса, через который следует передать кадр канального уровня, используются

значения адресов канального уровня (MAC-адресов), содержащиеся в заголовке кадра. Расчет ведется по следующей формуле:

`(MAC_отправителя XOR MAC_получателя) % число_физических_интерфейсов = индекс_физического_интерфейса`

Здесь `XOR` – побитовая операция, а `%` - операция получения остатка от деления. Значения индекса физического интерфейса для передачи начинаются с нуля.

В случае недоступности части физических интерфейсов передача идёт только по активным интерфейсам (то есть изменяется число физических интерфейсов), а индексы самих интерфейсов пересчитываются, сохраняя изначальный порядок в конфигурации и нумерацию с нуля.

Рассмотрим систему с настроенным виртуальным сетевым интерфейсом `bond` в режиме `balance-xor`, который включает в себя 4 физических сетевых интерфейса с индексами от 0 до 3. Этой системе поступает несколько кадров для передачи. MAC-адрес отправителя равен `24:01:C7:A3:8B:7E`, а MAC-адреса получателей указаны ниже.

```
00:24:1E:6D:FC:7D
44:45:53:36:A0:88
78:84:3C:94:06:A3
00:22:4C:FA:BE:C9
44:45:53:DC:E6:7A
68:76:4F:9A:99:FF
E0:0C:7F:53:C5:09
44:45:53:7C:70:65
BC:6E:64:3C:AD:EF
A4:5C:27:35:CE:7C
44:45:53:75:DA:1F
9C:5C:F9:74:17:50
00:09:BF:10:AD:FD
44:45:53:74:2A:F4
00:EB:2D:38:D0:F2
00:27:09:00:AC:1B
44:45:53:DC:53:CE
68:76:4F:18:EC:5C
64:B5:C6:58:97:DB
44:45:53:60:96:FE
00:1E:45:9C:9C:FF
E8:DA:20:CF:BB:CB
44:45:53:18:37:68
4C:21:D0:35:31:22
00:1F:C5:A3:8B:43
44:45:53:0B:25:65
00:21:9E:91:53:6B
00:26:59:B4:AE:9A
44:45:53:76:95:14
30:17:C8:8A:84:15
```

После передачи определённого количества кадров произошёл обрыв сети, из-за чего одновременно стали недоступны два физических интерфейса из четырёх, и оставшиеся кадры передавались по одному из двух активных интерфейсов. Известно, что по интерфейсу с исходным индексом 0 было передано 7 кадров, по интерфейсу с исходным индексом 1 – 12 кадров, по интерфейсу с исходным индексом 2 – 6 кадров, по интерфейсу с исходным индексом 3 – 5 кадров. Определите, какие из интерфейсов стали недоступны и сколько кадров было передано до обрыва канала. В ответе укажите три числа через пробел: номера интерфейсов (0, 1, 2 или 3) в порядке возрастания и количество переданных кадров. Если есть несколько вариантов того, какие интерфейсы могли стать недоступны, выберите любой вариант. Если есть несколько вариантов того, сколько кадров могло быть передано до обрыва, выберите максимальное число.

Ответ: 2 3 25

Решение:

Для решения задачи нам нужно посчитать побитовое исключающее ИЛИ для 30 пар 48-битных чисел и потом взять остаток от деления на количество активных интерфейсов. Можно заметить, что активных физических интерфейсов в любой момент времени либо 4, либо 2. Это позволяет нам посчитать исключающее ИЛИ только для последних 2 бит чисел пары и потом брать остаток от деления. Вычислим выражение из условия для каждой пары:

Адрес	Индекс интерфейса
`00:24:1E:6D:FC:7D`	3
`44:45:53:36:A0:88`	2
`78:84:3C:94:06:A3`	1

00:22:4C:FA:BE:C9	3
44:45:53:DC:E6:7A	0
68:76:4F:9A:99:FF	1
E0:0C:7F:53:C5:09	3
44:45:53:7C:70:65	3
BC:6E:64:3C:AD:EF	1
A4:5C:27:35:CE:7C	2
44:45:53:75:DA:1F	1
9C:5C:F9:74:17:50	2
00:09:BF:10:AD:FD	3
44:45:53:74:2A:F4	2
00:EB:2D:38:D0:F2	0
00:27:09:00:AC:1B	1
44:45:53:DC:53:CE	0
68:76:4F:18:EC:5C	2
64:B5:C6:58:97:DB	1
44:45:53:60:96:FE	0
00:1E:45:9C:9C:FF	1
E8:DA:20:CF:BB:CB	1
44:45:53:18:37:68	2
4C:21:D0:35:31:22	0
00:1F:C5:A3:8B:43	1
44:45:53:0B:25:65	3
00:21:9E:91:53:6B	1
00:26:59:B4:AE:9A	0
44:45:53:76:95:14	2
30:17:C8:8A:84:15	3

Дальше для рассмотрения возьмём вариант 1.

Нетрудно заметить, что в случае исправности всех интерфейсов в течение всего времени работы получилось бы следующее распределение кадров по интерфейсам:

Индекс	Количество (теор.)	Количество (факт.)
0	6	7
1	10	12
2	7	6
3	7	5

Видно, что интерфейсы с индексами 0 и 1 обработали больше кадров, а с индексами 2 и 3 - меньше, соответственно, первые два числа в ответе - 2 и 3.

Осталось понять, после какого кадра произошёл обрыв, для этого нужно посмотреть, когда мы передадим 1 лишний кадр через интерфейс 0 и 2 лишних кадра через интерфейс 1. Начнём пересчитывать номера интерфейсов ещё раз, но с конца:

Адрес	Индекс исходного интерфейса до обрыва	Индекс исходного интерфейса после обрыва	Не на тот интерфейс?
…	…	…	…
4C:21:D0:35:31:22	0	0	Нет
00:1F:C5:A3:8B:43	1	1	Нет
44:45:53:0B:25:65	3	1	Да
00:21:9E:91:53:6B	1	1	Нет
00:26:59:B4:AE:9A	0	0	Нет
44:45:53:76:95:14	2	0	Да
30:17:C8:8A:84:15	3	1	Да

Видно, что до обрыва должно быть передано максимум 25 кадров, чтобы получилась описанная в условии ситуация.

7. Технологии сортировки и фильтрации данных (1 балл) [Раз задача, два задача]

За две недели до олимпиады по информатике ответственный за систему её проведения, Николай Владимирович, собрал разработчиков, чтобы решить, какие ошибки нужно исправить и какие новые возможности нужно добавить. По результатам совещания ответственный нарисовал диаграмму Ганта:

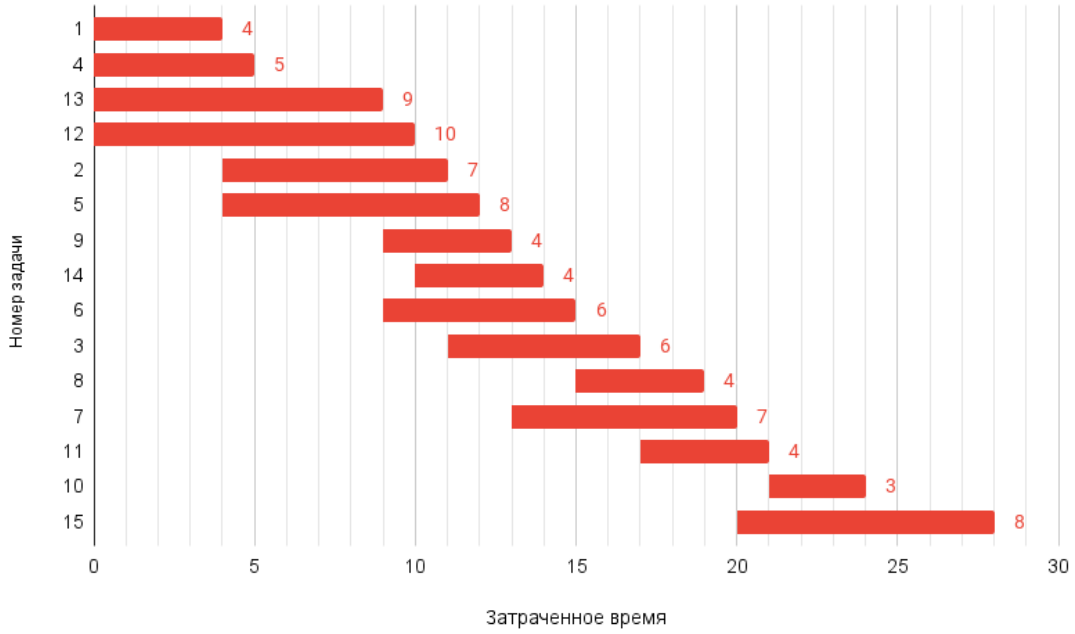


Диаграмма Ганта - это разновидность гистограммы, на которой показывается порядок и продолжительность выполнения поставленных задач. Каждая полоса на диаграмме показывает, когда начинается и заканчивается выполнение задачи. Диаграмма Ганта также показывает зависимые задачи, которые нужно выполнить перед тем, как будет выполнена основная задача. Каждая задача может не иметь зависимостей (и тогда она начинает выполняться в начальный момент времен) либо иметь одну или несколько зависимостей (и тогда она начинает выполняться сразу после выполнения последней из задач, от которых она зависит). Ответственный за проведение олимпиады не успел указать, какие задачи от каких зависят, и из-за этого на диаграмме можно однозначно увидеть только зависимость от этой последней задачи.

На каждой полосе для удобства отображается время в часах, которое необходимо для полного выполнения соответствующей задачи. Это время называется **сложностью задачи**. Отдел разработки достаточно большой, чтобы при необходимости выполнять параллельно все задачи, которые можно распараллелить в данный момент, при этом над одной задачей всегда работает один разработчик.

После собрания разработчики перенесли поставленные задачи в систему управления и приступили к их выполнению. В процессе разработки выяснилось, что ровно одна задача, которая изначально не имела зависимостей, оказалась зависимой от другой задачи, и это повлияло на общий срок выполнения всех задач.

Система управления отображает прогресс выполнения задач с помощью диаграммы сгорания, и в итоге получилась следующая диаграмма:

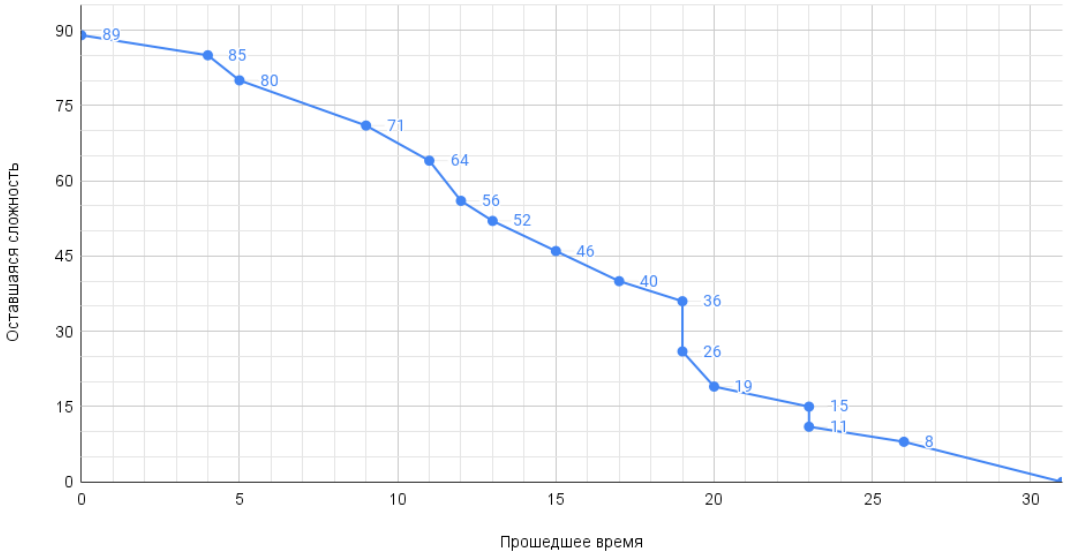


Диаграмма сгорания - это разновидность графика, на котором каждая точка ломаной на графике показывает момент изменения общей оставшейся сложности. Общая оставшаяся сложность считается как сумма сложностей всех оставшихся задач, в том числе выполняемых на текущий момент (без учёта степени их выполнения). Если в какой-то момент времени было выполнено несколько задач, соответствующему значению на оси абсцисс будет соответствовать несколько значений на оси ординат.

По этим двум диаграммам Николай Владимирович смог определить, какую зависимость он не учёл на своей диаграмме Ганта. Попробуйте и Вы это сделать. В ответе укажите два числа через пробел: номер задачи, у которой появилась зависимость, и номер задачи, от которой она стала зависеть.

Ответ: 12 13

Решение:

Решение можно свести к тому, чтобы построить диаграмму Ганта по диаграмме сгорания и сравнить её с диаграммой из условия. Задача облегчается тем, что мы добавляем зависимость к задаче, которая изначально ни от каких задач не зависела, то есть первое число - это 1, 4, 12 либо 13.

Построим таблицу, в которой отметим, в какой момент времени началась и завершилась задача, а также время выполнения (сколько единиц сложности “сгорело”) без указания номеров самих задач. Из диаграммы можно увидеть только конец и длительность задач, а из них определяется начало:

Начало	Конец	Сложность
0	4	4
0	5	5
0	9	9
4	11	7
4	12	8
9	13	4
9	15	6
11	17	6
15	19	4
9	19	10
13	20	7
19	23	4
19	23	4
23	26	3
23	31	8

Видно, что в начальный момент времени не начала выполняться задача со сложностью 10, то есть задача 12. Она начала выполняться в момент времени 9, и в этот же момент завершилась задача со сложностью 9, которая начала выполняться в начальный момент времени, то есть задача 13. Тогда получается, что у задачи 12 появилась зависимость в виде задачи 13.

8. Технологии программирования (3 балла)
[Робот-курьер]

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

Это интерактивная задача. Ваше решение должно взаимодействовать с программой-интерактором по описанному ниже протоколу.

На вас возложили ответственную задачу по управлением роботом-курьером. Карта, по которой перемещается робот, представляет из себя поле размера $n \times m$ (n строк и m столбцов). Каждая клетка поля может быть либо тротуаром (‘.’), либо проезжей частью дороги (‘+’).

Ровно k клеток проезжей части содержат регулируемые пешеходные переходы. Гарантируется, что для любого пешеходного перехода **ровно две** противоположные относительно него соседние с ним по стороне клетки (то есть верхняя и нижняя или левая и правая) являются клетками тротуара. Соответственно, с обоих концов каждого пешеходного перехода расположен светофор.

Робот может перемещаться только по тротуарам и пешеходным переходам. Для определения цвета светофора робот обладает камерой с разрешением $h \times w$ (где w четно). Для управления роботом вы можете передавать ему следующие команды:

- «turn c», где $c \in$ {‘L’, U‘, R’, ‘D’}, означает поворот в соответствующую сторону (влево, вверх, вправо или вниз);
- «move» означает перемещение на одну клетку вперед относительно текущего направления;
- «camera» означает получение изображения с камеры; в ответ на эту команду вы получаете таблицу из $h \times w$ символов, каждый из которых описывает преобладающий цвет (‘r’, ‘g’ или ‘b’ — красный, зеленый или синий) в соответствующей области пространства перед роботом;
- «wait t» означает ожидание в течение t секунд.

На поворот или перемещение требуется ровно одна секунда. Получение изображения с камеры времени не требует. Каждый светофор горит одним цветом в течение фиксированного периода времени, после чего моментально переключается и горит другим цветом то же время (и так далее). Этот период времени вам неизвестен и может быть разным у разных светофоров, однако гарантируется, что он не превышает 10^6.

Светофоры могут находиться на разной высоте и на разном расстоянии сбоку от соответствующего перехода. Если робот находится около перехода с i-м светофором и смотрит в его направлении, на изображении с камеры светофор будет занимать две клетки в a_i-й и $(a_i + 1)$-й снизу строках в столбце на расстоянии b_i от центра (слева от центра, если $b_i < 0$, и

справа, если $b_i > 0$). Для светофора, горящего красным, нижняя из этих двух клеток равна ‘b’, а верхняя равна ‘r’. Для зеленого светофора нижняя клетка равна ‘g’, а верхняя — ‘b’. Остальные клетки на изображении могут любого из трех цветов.

Требуется переместить робота из клетки (i_1, j_1) (i_1-я сверху строка, j_1-й слева столбец) в клетку (i_2, j_2). Начинать пересекать пешеходные переходы можно только если на соответствующем светофоре горит зеленый сигнал. Если движение по переходу начато, когда на светофоре горит зеленый сигнал, можно считать, что как минимум в течение еще двух секунд находиться на переходе безопасно, то есть можно гарантированно переместиться на тротуар на противоположной стороне дороги.

Напишите программу, сообщающую роботу команды, безопасно приводящие его из стартовой клетки в конечную. Минимизировать затраченное в пути время не требуется. В изначальной клетке робот находится в направлении «вверх» (‘U’).

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода дано единственное целое число t — количество наборов входных данных в тесте ($1 \le t \le 50$).

Наборы входных данных поступают вам на ввод последовательно: для каждого набора сначала вводится информация о карте и роботе, а затем сразу запускается протокол взаимодействия с интерактором (см. ниже).

В первой строке описания карты даны два целых числа n и m — размеры карты ($1 \le n, m \le 50$). Следующие n строк содержат по m символов каждая и описывают карту. Символ на j-й позиции i-й строки описывает клетку с координатами (i, j) и равен ‘.’, если это клетка тротуара, и ‘+’, если это клетка проезжей части.

В следующей строке даны три целых числа k, h и w — количество переходов со светофорами и разрешение камеры, соответственно ($k \le n \cdot m$; $2 \le h, w \le 8$; w четно).

Следующие $3k$ строк описывают светофоры: по три на каждый из k переходов. В первой строке для i-го светофора дано положение соответствующего ему перехода (r_i, c_i) ($1 \le r_i \le n$; $1 \le c_i \le m$). Во второй строке дано описание светофора с одного из двух концов перехода в формате «$d_{i,1}\ a_{i,1}\ b_{i,2}$», где d_1 указывает на направление перехода, соответствующее этому светофору, а $a_{i,1}$ и $b_{i,1}$ — его высота и расстояние от центра перехода, соответственно ($d_{i,1} \in$ {‘L’, U‘, R’, ‘D’}; $1 \le a_{i,1} < h$; $1 \le |b_{i,1}| \le \frac{w}{2}$). В третьей строке в том же формате описывается светофор с противоположной стороны перехода.

Наконец, в последней строке набора входных данных даны четыре целых числа i_1, j_1, i_2 и j_2 — координаты стартовой и конечной клеток, соответственно ($1 \le i_1, i_2 \le n$; $1 \le j_1, j_2 \le m$).

Гарантируется, что все (r_i, c_i) различны, а описания светофоров корректны: направления $d_{i,1}$ и $d_{i,2}$, указанные во вводе, противоположны и соответствуют направлениям, в которых от этого перехода расположен тротуар. Также гарантируется, что конечная клетка достижима из стартовой.

После прочтения вашей программой входных данных для очередного набора запускается протокол взаимодействия с интерактором.

Протокол взаимодействия

Взаимодействие с интерактором заключается в сообщении вашей программой интерактору очередного действия робота, после чего интерактор будет выводить результат выполнения этого действия. Для большего понимания обратите внимание на пример теста в условии.

Чтобы повернуть робота, выведите «turn c», где $c \in$ {‘L’, U‘, R’, ‘D’}. В результате выполнения этого действия робот повернется «лицом» в соответствующем направлении, и интерактор выведет «OK» на отдельной строке.

Чтобы переместить робота, выведите «move». В таком случае робот переместится на одну клетку вперед в том направлении, в котором он повернут. Если это действие успешно, интерактор выведет «OK» на отдельной строке. Если при этом робот достиг конечной клетки (i_2, j_2), интерактор перейдет к рассмотрению следующего набора входных данных и подаст соответствующие входные данные на ввод вашей программе (либо завершится и засчитает ваше решение, если это был последний набор входных данных).

Если же робот при таком перемещении попадает в непроходимую клетку, выходит за пределы карты или выезжает на пешеходный переход на красный свет, интерактор выведет «FAIL» и завершится с вердиктом Wrong Answer. Во избежание получения некорректного вердикта, считав «FAIL», ваше решение также должно завершиться.

Чтобы сделать снимок, выведите «camera». В ответ интерактор выведет h строк по w символов каждая. Каждый символ равен ‘r’, ‘g’ или ‘b’ и задает цвет соответствующего «пикселя». Если непосредственно перед роботом не находится пешеходный переход, все символы будут случайными. Если же робот стоит у перехода, то два символа, соответствующие положению на «изображении» светофора напротив, будут отражать цвет этого светофора как описано в условии.

Чтобы подождать t секунд ($1 \le t \le 2 \cdot 10^6$), выведите «wait t». В ответ интерактор выведет «OK» на отдельной строке и обновит состояние всех светофоров, цвет которых за это время поменяется.

Запрещается делать более 25 команд ожидания в одной и той же клетке поля. Если ваше решение совершает хотя бы 26 запросов ожидания из одной и той же клетки, интерактор в ответ выведет «FAIL» и завершится с вердиктом Wrong Answer.

Пример

Стандартный ввод	**Стандартный вывод**
2 2 3	

Стандартный ввод	Стандартный вывод
.++ ... 0 2 2 1 1 2 3 OK OK OK OK OK 3 3 .+. .++ .+. 2 3 4 1 2 L 1 -1 R 2 -2 2 3 U 2 -1 D 1 2 3 1 3 3 OK OK OK bbbb gbbb bbbb OK OK OK bbbb bbbr bbbb OK bbbb bbbb bbbg OK OK	turn D move turn R move move move camera move move turn D camera wait 10 camera move move

Замечание

Вывод каждой команды ваша программа должна завершать выводом символа перевода строки (endl, ‘\ n’) и сбросом буфера вывода. Сбросить буфер можно с помощью

«fflush(stdout)» в C и C++, или «cout.flush()» только в C++,

«System.out.flush()» в Java,
«sys.stdout.flush()» в Python,
и «Console.Out.Flush()» в C#.

В Pascal и Delphi сброс буфера при выводе в стандартный поток вывода происходит автоматически.

Решение, не выполняющее эти действия, может получить произвольный вердикт (скорее всего, Time Limit Exceeded или Idleness Limit Exceeded).

В примере из условия добавлены лишние пустые строки, чтобы проиллюстрировать взаимодействие с интерактором и порядок ввода-вывода при этом взаимодействии. В решении выводить эти пустые строки не надо.

Пояснение к примеру

В клетке $(3,2)$ нет перехода, поэтому по ней нельзя перемещаться;

При пересечении перехода в $(1,2)$ в направлении ‘R’ светофор находится на высоте 2 и на расстоянии -2 от центра: соответственно, его клетки на изображении располагаются в первом столбце во второй и третьей снизу строках. Для данного снимка они равны ‘b’ в верхней строке и ‘g’ во второй, поэтому его сразу можно пересекать.

При пересечении перехода в $(2,3)$ в направлении ‘D’ светофор находится на высоте 1 и на расстоянии 2 от центра, то есть в четвертом столбце в двух нижних строках. На первом изображении он горит красным, а после ожидания («wait 10») — зеленым.

Решение:

Это задача на реализацию. Основными идейными сложностями являются поиск пути и преодоление светофора за не более чем 25 запросов ожидания.

Для поиска пути можно было использовать dfs или bfs (поиск в глубину или поиск в ширину). Решения на Python, использовавшие dfs, могли столкнуться с трудностями, связанными с неэффективностью рекурсии или с недостаточным стеком рекурсии, поэтому лучше было сразу писать поиск в ширину.

Перечислим, как удобно хранить данные в программе.

- Будем хранить dir - текущее направление. В варианте задачи с поворотами типа «rotate» стоило упорядочить все направления против часовой стрелки, чтобы можно было находить угол поворота через разницу их позиций в этой последовательности. Также удобно для каждого направления в dict или map хранить соответствующие ему перемещения по каждой из двух координат.
- Также заведем словарь/ассоциативный массив (dict или map) lights, сопоставляющий трой- кам (r, c, LR) информацию о светофорах, расположенных рядом с переходами в соответствую- щих клетках. Здесь (r, c) задает позицию перехода, а LR - флаг, отвечающий за направление (true для направлений «влево» и «вверх», и false иначе).

Тогда весь алгоритм заключается в следующем: найдем с помощью поиска в ширину путь из стартовой клетки в конечную по проходимым клеткам, то есть по клеткам тротуара и клеткам, соответствующие которым ключи есть в lights, а затем, двигаясь по этому пути, будем убеждаться, что каждый переход безопасен.

Для поиска в ширину достаточно, доставая очередную клетку из очереди, перебирать всех ее соседей (то есть перебирать все четыре направления движения и проверять, что соответствующее направление движения не выводит робота за пределы карты), и для каждой соседней клетки, в которой роботу разрешено находиться, обновлять до нее расстояние.

Затем, чтобы двигаться по найденному пути, будем поддерживать текущее положение робота (r, c) и его направление движения dir, и для каждой следующей клетки выполнять следующие действия.

1. Если клетка расположена не в направлении dir от текущей, выполним поворот и обновим направление.
2. Если соответствующая клетка есть в lights, выполним запрос к камере.
3. Пусть светофор в текущем направлении, то есть lights[(r, c, dir $\in$ {‘L’, ‘U’})], задается числами a_i и b_i. Тогда посмотрим на ячейку изображения в строке $h - a_i$ и столбце $w + b_i$, и определим цвет светофора.
4. Пока светофор красный, будем ждать какое-то время и повторять запрос к камере.
5. Как только светофор стал зеленым, либо если следующая клетка в принципе была клеткой тротуара, совершим перемещение вперед.

Положение нужного пикселя на изображении могло быть в немного других индексах (на ± 1 отличающихся от указанных выше) в завсимости от вашего варианта задачи и знака числа b_i.

Осталось только понять, как оптимально ждать зеленый сигнал светофора. Ждать по 1 секунде не получится, так как может оказаться, что за 25 секунд светофор не переключится. Но здесь есть две хорошие стратегии, которые позволяют уложиться в ограничение на число запросов: ждать случайное время или ждать каждый раз в два раза больше, чем в предыдущий.

Если ждать случайное время, то с каждым ожиданием будет вероятность 0.5 «попадания» в нужный цвет, что дает вероятность меньше 10^{-6} получить красный хотя бы 20 раз подряд. Ожидание со временами, равными степеням двойки, позволяет гарантировать, что мы не пропустим следующее переключение светофора (так как если предыдущих ожиданий не хватило для переключения, то следующим не получится «перескочить» через целый период).

9. Технологии программирования (3 балла)

[Скрещивание]

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

Вы — владелец зоопарка, в котором сейчас обитают n редких экзотических видов змей, i-й из которых имеет *опасность* a_i.

Держать животных в неволе вам не очень хочется, но и выпускать опасных змей в общее пространство тоже, разумеется, нельзя. Для исправления этой проблемы вы решили скрестить некоторые виды, от чего их экзотичность меньше не станет, а вот опасность может уменьшиться. Для одного скрещивания вы можете выбрать два вида змей под номерами i и j, и скрестить их в новый вид, имеющий опасность $a^*_{i,j} = a_i \oplus a_j$, где за $\oplus$ обозначена операция побитового исключающего «ИЛИ» (также обозначается как xor или ^).

Чтобы не получать слишком похожие виды, каждый из имеющихся n видов может участвовать только в одном скрещивании. Также невозможно скрестить два вида, хотя бы один из которых уже является результатов скрещивания каких-то из изначальных n видов. Иными словами, скрещивать можно только исходные виды, и только по парам.

В конце вы получаете новый набор видов, в который войдут исходные виды, не поучаствовавшие в скрещиваниях, а также все результаты скрещиваний. То есть исходные виды, которые были скрещены с какими-то еще, в этот набор не войдут (опять же, потому что нет смысла отводить в зоопарке отдельное место под несколько близких видов — посетители все равно не увидят разницу).

Определите, какие пары видов змей следует скрестить, чтобы максимум из опасностей полученного набора видов был как можно меньше.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода дано единственное целое число t — количество наборов входных данных в тесте ($1 \le t \le 50$). Далее следуют сами наборы входных данных.

В первой строке набора входных дано целое число n — количество видов экзотических змей в наличии изначально ($1 \le n \le 50\,000$). Гарантируется, что сумма n по всем наборам входных данных не превосходит 10^5.

Во второй строке перечислены n целых чисел a_i — значения опасности этих видов ($0 \le a_i \le 10^9$).

Формат выходных данных

Для каждого набора входных данных выведите в отдельной строке единственное целое число — минимальное возможное значение максимальной опасности, которое можно получить такими скрещиваниями.

Пример

Стандартный ввод	**Стандартный вывод**
4 3 1 2 3 7 1 4 9 16 25 36 49 6 0 1 2 5 6 7 10 5 7 14 10 12 2 1 13 3 11	1 21 4 8

Замечание

В первом примере выгодно скрестить 2 и 3, и получить исходный вид с $a_1 = 1$ и новый с $a_{2,3} = 1$.

Во втором примере скрещиваются 16 с 25 (получается $a_{4,5} = 9$) и 36 с 49 (получается $a_{6,7} = 21$).

В третьем примере скрещиваются 2 с 6 и 5 с 7.

Решение:

Пойдем с конца: пусть ответом на задачу является число x. Посмотрим на его битовую запись и найдем в ней первую (старшую) единицу lead(x). Пусть она стоит на позиции y с конца, то есть соответствует 2^y при разложении x в сумму степеней двойки. Тогда заметим, что любой a_i, у которого lead(a_i) > y, то есть старшая единица стоит раньше, должен быть поставлен в пару с a_j, у которого все биты старше y совпадают с a_i, чтобы их xor давал в старших битах 0.

Таким образом, чтобы в ответе lead был равен y, все a_i с lead(a_i) > y должны разбиваться на пары с одинаковым $\text{head}_{y+1}(a_i) = a_i >> (y+1)$. Здесь за >> обозначен битовый сдвиг вправо, то есть операция, отбрасывающая последние биты числа. В данном случае $a_i >> (y+1)$ отбрасывает все биты с нулевого по y-й включительно.

Алгоритм получается следующий: переберем y от 29 до 0 (можно вместо перебора сделать двоичный поиск, чтобы немного увеличить эффективность), сгруппируем все a_i по их head_y, то есть по $30 - y$ старшим битам, и проверим, что все группы, у которых head_y ненулевой, содержат четное число элементов. Если это так, то можно добиться того, чтобы

во всех $30 - y$ старших битах у каждого числа в ответе стояли нули, просто сгруппировав исходные числа по парам внутри групп.

Как только мы нашли такой y, для которого хотя бы одна из групп оказалась нечетного размера, понятно, что в ответе в соответствующем бите будет стоять 1. При этом по всем head_{y-1} группы все еще четного размера, поэтому числа надо разбивать на пары внутри таких групп. Тогда сгруппируем a_i по их head_{y-1} и внутри каждой группы выделим те, у которых следующий бит равен 0 и те, у которых следующий бит равен 1.

После этого для каждой группы решим задачу независимо.

Если количество тех, у которых следующий бит равен 1, четно, то их можно внутри этой группы разбить на пары так, чтобы все результирующие числа имели в следующем бите 0, и в таком случае они все будут меньше ответа.

Иначе можно разбить на пары все, кроме одного. А еще одно число либо оставить как есть, и тогда ответ для этой группы просто будет равен минимальному из чисел с единицей в сле- дующем бите, либо сопоставить ему в пару число из той же группы, но с нулем в следующем бите.

В таком случае достаточно найти $\min_{\substack{a_i \in group0 \\ a_j \in group1}} a_i \oplus a_j$,

где group0 и group1 - элементы текущей группы с нулем или единицей в следующем бите, соответственно.

Поиск такого минимума является классической задачей, которая решается с помощью битового бора: будем воспринимать каждое число как строку из 30 нулей и единиц, добавим все элементы одной подгруппы в такой бор, а для каждого элемента второй подгруппы будем искать ему пару, минимизирующую их xor, стараясь каждый раз в боре идти по совпадающему биту.

Теперь, когда в каждой группе получено минимально возможное максимальное число, ответом будет максимум из ответов для всех групп. Общее время работы такого решения - O(n log A), где A - ограничение сверху на значения a_i.

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

1. Кодирование информации. Системы счисления (3 балла) [Утроения]

Петя научился переводить запись натурального числа в другую систему счисления и для тренировки переводил числа из десятичной в двоичную систему счисления. Он заметил, что для некоторых чисел выполняется следующее условие: количество разрядов в записи такого числа в двоичной системе счисления ровно в три раза больше количества разрядов в записи этого же числа в десятичной системе счисления. Петя случайно нашел два таких числа: $6_{10}=110_2$ и $300_{10}=100101100_2$, но уверен, что таких чисел больше, и решил подойти к их поиску системно. Он написал программу, последовательно перебирающую все натуральные числа и добавляющую в список очередное число, если оно удовлетворяет указанному выше условию, и запустил её на суперкомпьютере. Какое число оказалось в списке под номером 10^8-1? В ответе укажите целое число.

Примечание: числа в списке Пети нумеруются с 1.

Ответ: 1057806682

2. Кодирование информации. Системы счисления (2 балла) [Дроби]

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.

Ответ: 28/85

3. Кодирование информации. Количество информации. Кодирование текста (1 балл) [Программируемый станок]

Программируемый станок умеет выполнять N различных операций. В базовой прошивке программа для станка сохранялась как последовательность номеров операций, причем каждый номер операции записывался в памяти с использованием минимально возможного, одинакового для всех номеров количества бит. При этом программа из 256 операций занимала ровно всю доступную для записи программы память станка.

Вася решил переписать прошивку, чтобы увеличить максимальную длину программы. Он заметил, что из-за технологических ограничений станок может исполнить только программу, количество операций в которой кратно трем. Тогда любая разрешенная программа является последовательностью комбинаций из трех операций, причем, количество таких комбинаций, которые станок может выполнить в восемь раз меньше, чем количество всех комбинаций, которые можно составить из трех операций, которые умеет выполнять станок. Тогда Вася написал код новой прошивки так, чтобы программа для станка сохранялась в памяти как последовательность номеров только разрешенных комбинаций, причем каждый номер комбинации записывался в память с использованием минимально возможного, одинакового для всех номеров количества бит. После изменения прошивки появилась возможность записывать в память станка коды комбинаций, соответствующие программе максимальной длиной в 318 операций.

При каком максимальном значении N это возможно? В ответе укажите целое число.

Ответ: 32

4. Кодирование информации. Количество информации (2 балла)

[Апрель]

Таня готовится к ЕГЭ по информатике и решает задачу, связанную с траекторией вычислений. Задача звучит следующим образом:

Исполнитель Апрель преобразует число на экране. У исполнителя есть три команды:

1. *Прибавить 1.*
2. *Умножить на 3 и поделить нацело на 2.*
3. *Умножить на 2*

Программа для исполнителя Апрель - это последовательность команд. Сколько существует программ, для которых при исходном числе 10 результатом является число 5094 и при этом траектория вычислений программы содержит число 51?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы 213 при исходном числе 8 траектория будет состоять из чисел 12, 13, 26.

Петя сказал, что нельзя серьезно научиться информатике, решая давно известные задачи, и предложил дополнить условие. Пусть все возможные различные траектории вычислений из указанной задачи равновероятны, и известны три сообщения относительно наугад выбранной траектории:

1. Траектория вычислений программы содержит числа 19 и 33.
2. Траектория вычислений программы содержит числа 18 и 35.
3. Траектория вычислений программы содержит числа 17 и 31.

Необходимо упорядочить эти три сообщения по возрастанию количества собственной информации в каждом из этих сообщений. В ответе запишите номера сообщений в нужном порядке без пробелов. Если некоторые сообщения содержат в себе одинаковое количество информации, укажите их номера в порядке возрастания.

Ответ: 312

5. Основы логики. Анализ логических функций (3 балла)

[Сплошные следования]

Известно, что логическое высказывание $x \to y$ является истинным, а логическое высказывание $y \leftrightarrow z$ является ложным. Выберите среди перечисленных ниже высказываний все те, для которых в этом случае можно однозначно определить их логическое значение (истинность или ложность).

1. $(x \to y) \leftrightarrow z$
2. $\big((x \wedge \bar{y}) \leftrightarrow (y \wedge z)\big) \to (y \leftrightarrow z)$
3. $(x \to y) \to (y \; xor \; z)$
4. $(\overline{z \; xor \; y}) \to (y \to z)$
5. $(\bar{y} \wedge z) \to (y \leftrightarrow \bar{x})$

Ответ: 2, 3, 4

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

[Затухание]

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

$$\Big(\Big(\big((\bar{A} \wedge B) \leftrightarrow \bar{C}\big) \to C\Big) \to \Big(\big((\bar{B} \wedge C) \leftrightarrow \bar{D}\big) \to D\Big)\Big) \to \Big(\big((\bar{C} \wedge D) \leftrightarrow \bar{E}\big) \to E\Big)\Big) \to \Big(\big((\bar{D} \wedge E) \leftrightarrow \bar{F}\big) \to F\Big)$$

*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: D or not E or F || D or F or not E || not E or D or F || not E or F or D || F or D or not E || F or not E or D

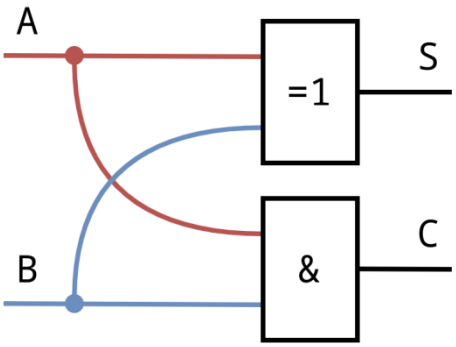
7. Основы логики. Синтез выражения по логической схеме (2 балла)

[Полусумматоры]

Витя увлекается электротехникой и собирает различные логические схемы. На последнем занятии кружка электроники и электротехники руководитель кружка, Николай Александрович, рассказал про полусумматор.

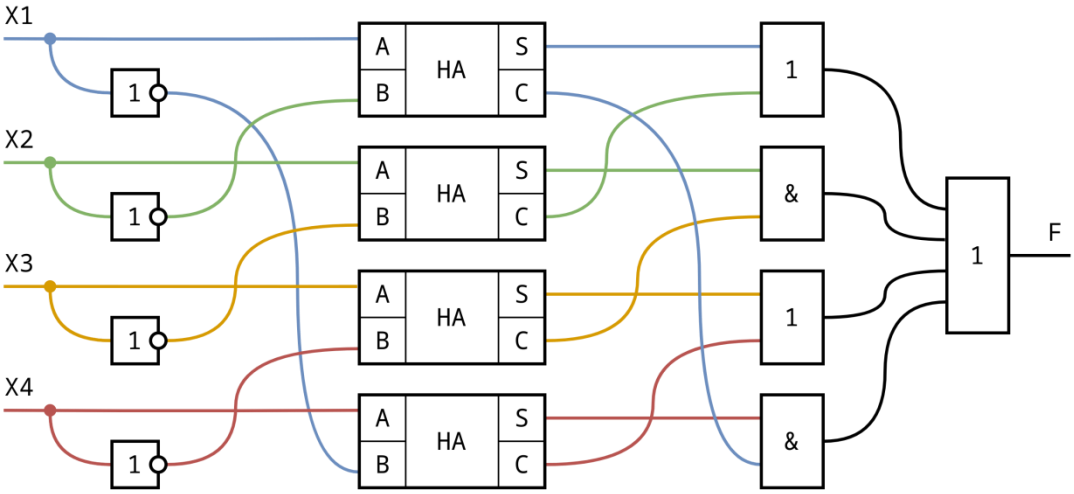
Полусумматор – это логическая схема, которая принимает на вход два логических значения A и B и выдаёт на выходе два логических значения S и C (сумму A и B с учётом переноса). Ниже представлена таблица истинности для такой схемы и пример её реализации с использованием исключающего ИЛИ и И:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



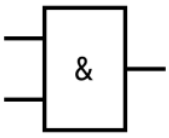
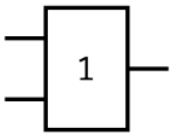
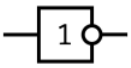
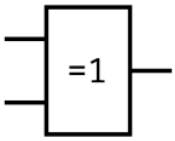
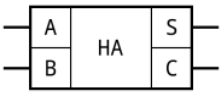
Часто полусумматор обозначают на схемах как отдельный элемент.

Оставшись после занятия, Витя решил собрать свою схему с использованием нескольких полусумматоров:



Николай Александрович увидел схему и сказал, что эта схема будет выдавать ложь только для одной комбинации значений переменных. Найдите эту комбинацию и в ответе укажите подряд четыре значения 0 или 1, соответствующие значениям логических переменных в порядке возрастания их индексов, где 0 означает ложное значение, а 1 – истинное значение. Если таких комбинаций несколько, укажите любую из них. Если таких комбинаций нет, укажите в ответе *NULL*. Пример записи ответа: 1101.

Примечание: на схеме используются следующие обозначения логических элементов:

И (конъюнкция)	ИЛИ (дизъюнкция)	НЕ (инверсия)	Исключающее ИЛИ	Полусумматор
				

Цвета на схеме предназначены для упрощения чтения и не несут никакой дополнительной информации.

Ответ: 1010

8. Алгоритмизация и программирование. Формальный исполнитель (1 балл) [Последовательность]

Изначально последовательность чисел состоит из одного числа: [1].

Дальнейшие последовательности чисел формируются по следующему алгоритму:

1. Посчитать сумму чисел в текущей последовательности и взять остаток от её деления на 13.
2. Дописать получившееся на предыдущем шаге число в начало последовательности.
3. Если количество чисел в последовательности стало равно N, остановить выполнение алгоритма, иначе перейти на шаг 1.

Например, при N=5 в результате выполнения алгоритма получится последовательность [8, 4, 2, 1, 1].

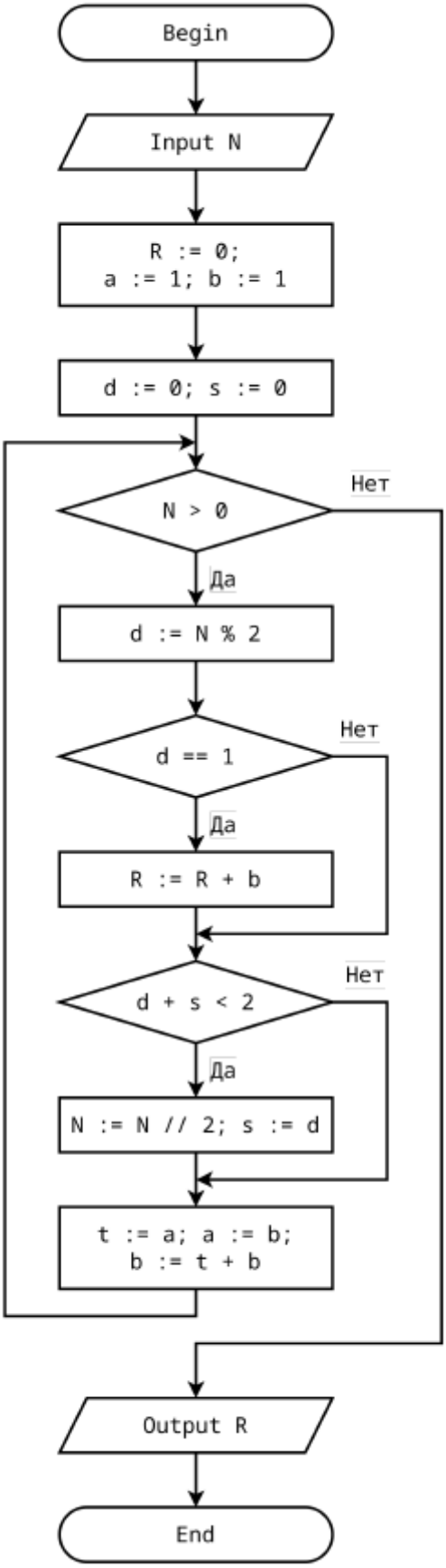
Алгоритм запустили с параметром $N=10^7-100$.

Определите и запишите в ответ через пробел два числа из получившейся в результате последовательности. Сначала число, стоящее на позиции 200, а затем число, стоящее на позиции 99999. Нумерация элементов последовательности идет слева направо с 1.

Ответ: 8 9

9. Алгоритмизация и программирование. Блок-схема (2 балл)
[Опять Фибоначчи]

Дана блок-схема алгоритма:



Определите, какое число нужно подать на вход, чтобы в результате было выведено число 272?
Если таких чисел несколько, укажите наибольшее из них.

Примечание: оператор A//B вычисляет частное от целочисленного деления A на B, оператор A%B вычисляет остаток от целочисленного деления A на B.

Ответ: 2184

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

Петя написал программу, которая заполняет квадратную матрицу последовательностью натуральных чисел, начиная с N сверху вниз по линиям, параллельным побочной диагонали матрицы от верхнего левого угла. Например, при N=1 первые 10 чисел заполнят матрицу следующим образом:

$$\begin{bmatrix} 1 & 2 & 4 & 7 & \dots \\ 3 & 5 & 8 & & \\ 6 & 9 & & & \\ 10 & & & & \\ \dots & & & & \end{bmatrix}$$

Матрица имеет неограниченный размер, и заполнены только элементы влево-вверх от побочной диагонали. Индексация элементов матрицы считается от верхнего левого угла, индексированного как (0,0). Первый индекс означает номер строки, а второй – номер столбца. Известно, что элемент с индексом (22, 19) равен 1000. Определите значение N. В ответе укажите целое число.

Ответ: 117

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

1. Электронные таблицы. Адресация ячеек и вычисления (1 балл) [3x3]

Дан фрагмент электронной таблицы:

	A	B	C	D	E
1	1	2	5	11	
2	3				
3	7				
4	13				
5					

В ячейку B2 поместили формулу вида =#A#1*#B#1*#A#2, где на месте каждого знака # может быть или не быть символ $.

Ячейку B2 скопировали во все ячейки диапазона B2:D4.

Определите конкретную формулу, которая была в ячейке B2, если известно, что в ячейке D4 получилось значение 11466. В ответе укажите последовательность из шести двоичных разрядов, означающих наличие (1) или отсутствие (0) символов $ на позициях, отмеченных знаком # считая слева направо. Например, ответ 111001 соответствует формуле =A1*$B1*A$2.

Если такой формулы не существует, укажите в ответе NULL.

Ответ: 101010

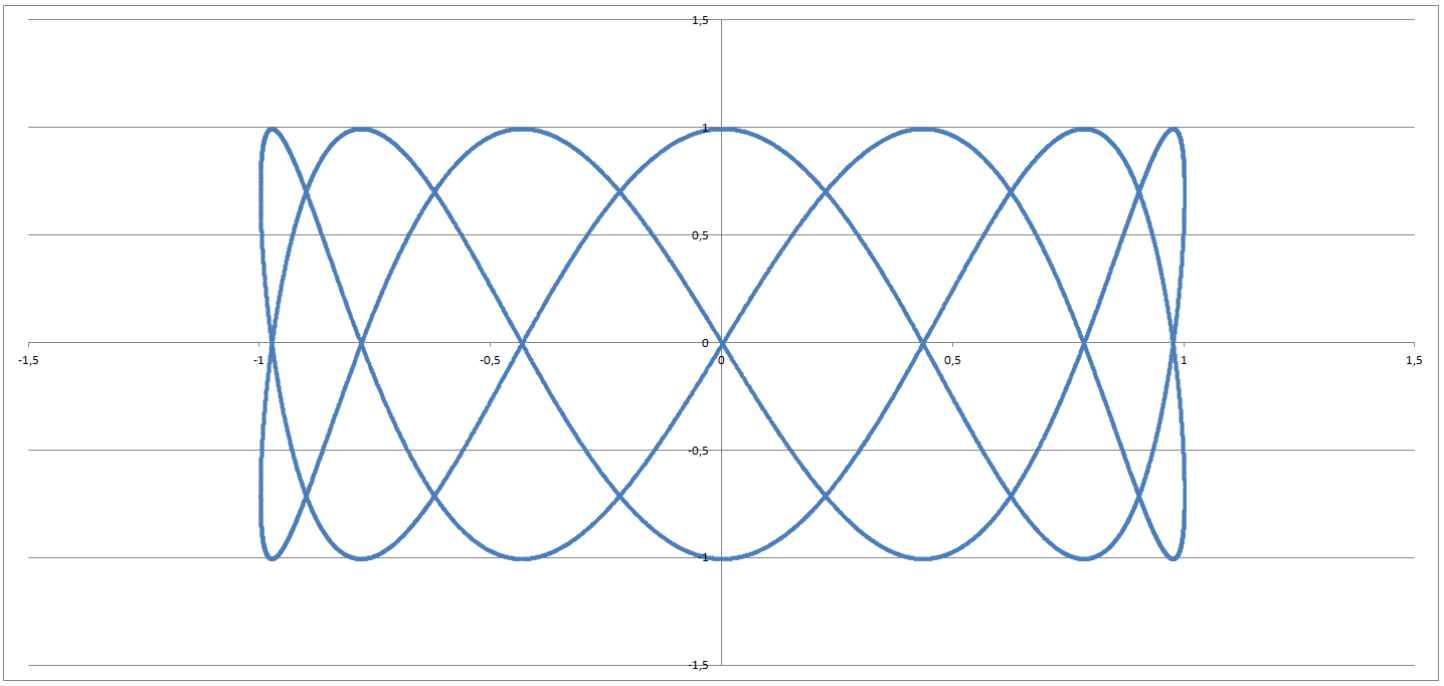
2. Электронные таблицы. Графики и диаграммы (2 балла) [Фора физикам]

Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D
1		[redacted]		
2	0	=SIN(B$1*A2)	=SIN(C$1*A2)	
3	0,001			
4	0,002			
5	0,003			
6	0,004			
7	0,005			
8	0,006			

Ячейки диапазона A2:A10000 заполнены числами, образующими арифметическую прогрессию с шагом 0,001.

Ячейку B2 скопировали во все ячейки диапазона B3:B10000. Ячейку C2 скопировали во все ячейки диапазона C3:C10000. Затем выделили диапазон B2:C10000 и построили по нему точечную диаграмму. Получился следующий результат:



Определите числа, которые находятся в ячейках B1 и C1, если про них известно следующее:

1. Оба числа являются натуральными.
2. Хотя бы одно число больше 1000.
3. Сумма этих чисел минимально возможная.

В ответе укажите через пробел сначала значение в ячейке B1 и затем значение в ячейке C1.

Ответ: 286 1001

3. Сортировка и фильтрация данных (2 балла)

[Продажи]

Таблица Orders в реляционной базе данных содержит записи о заказах, состоящие из значений трех атрибутов:

1. Customer – имя заказчика, может принимать значения только A, B или C.
2. Product – название продукта, может принимать только значения P1, P2 или P3.
3. Quantity – количество единовременно заказанного продукта конкретным заказчиком, может принимать любое целое положительное значение. Каждый заказчик может неограниченное количество раз заказывать некоторое количество любого продукта, о чем добавляется очередная запись в таблицу.

Известны результаты выполнения нескольких SQL запросов:

SELECT SUM(Quantity) FROM Orders GROUP BY Customer ORDER BY Customer	70 65 45
SELECT SUM(Quantity) FROM Orders GROUP BY Product ORDER BY Product	65 55 60
SELECT SUM(Quantity) FROM Orders WHERE Customer IN (A, B) GROUP BY Product ORDER BY Product	25 50 60
SELECT SUM(Quantity) FROM Orders WHERE Product=P1 and Customer=A	15
SELECT SUM(Quantity) FROM Orders WHERE Product=P3 and Customer=B	25

Что будет выведено в результате выполнения запроса:
SELECT SUM(Quantity) FROM Orders WHERE Product=P2 and Customer=A

Примечание. Семантика использованных операторов:

SELECT ###	Выводит значения атрибутов или функций, вычисленных от множества значений атрибутов, перечисленных после ключевого слова SELECT
FROM ###	Указывает название таблицы, из которой берутся значения
WHERE ###	Фильтр. В вывод SELECT, в том числе в качестве аргументов вычислимых функций, будут попадать только записи, удовлетворяющие условиям фильтра. Конструкция X IN (T1, T2, …, TN) означает, что фильтру удовлетворяют

	только записи, в которых атрибут X равен одному из перечисленных в скобках значений.
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.
SUM(###)	Функция, вычисляющая сумму 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.

Ответ: 20

4. Мультимедиа технологии (3 балла)

[HSB]

В компьютерной графике для растрового изображения часто строят гистограмму яркостей по каналам. Она представляется собой три графика для красного, зеленого и синего каналов соответственно, где на каждом графике по оси абсцисс отложены возможные значения яркости пикселей в канале изображения (слева направо от 0 до 255), а по оси ординат количество точек, имеющих такую яркость в выбранном канале.

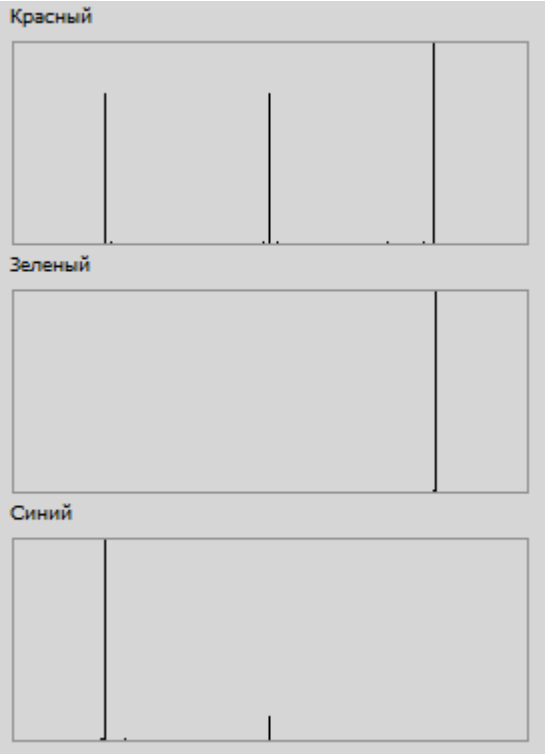
Известно, что исходное изображение состоит из 20 равных по количеству пикселей областей, окрашенных следующим образом:

Цвет	Количество областей на изображении
Красный=250, Зеленый=5, Синий=5	8
Красный=250, Зеленый=128, Синий=5	6
Красный=250, Зеленый=250, Синий=5	4
Красный=128, Зеленый=250, Синий=5	2

Распространенным инструментом цветокоррекции является изменение значение цветов пикселей в цветовой модели HSB. В этом случае для пикселей изображения могут применяться операции изменения цветового тона (Hue) от –180° до +180°, изменения насыщенности (Saturation) от –100 до +100 (в процентах), изменения яркости (Brightness) от –100 до +100 (в процентах). Пусть задан набор операций цветокоррекции, которые можно применять к изображению:

Номер операции	Операция
1	Hue +60°
2	Hue +120°
3	Hue +180°
4	Hue –60°
5	Hue –120°
6	Saturation –33
7	Saturation –67

Известно, что к исходному изображению последовательно применили ровно две операции, после чего получили следующие гистограммы по каналам:



Определите примененные операции и укажите их номера в ответе через пробел в порядке возрастания номеров.

Ответ: 1 6

5. Телекоммуникационные технологии (2 балла). [DDoS-атака]

В вычислительном центре университета стоит высокопроизводительный сервер, обрабатывающий запросы двух типов. Все принятые запросы становятся в единую бесконечно большую очередь на обработку. На определение типа запроса у сервера уходит 3 миллисекунды, если запрос оказался известного типа, он сразу же отправляется на обработку, в противном случае он игнорируется. Запрос типа 1 обрабатывается 7 миллисекунд, а запрос типа 2 обрабатывается 12 миллисекунд. Определение типа запроса и его обработка выполняются последовательно для каждого запроса. Это значит, что если в очереди сервера есть два запроса, то сначала будет определён тип первого запроса, затем при необходимости первый запрос будет обработан, и только потом будет определяться тип второго запроса.

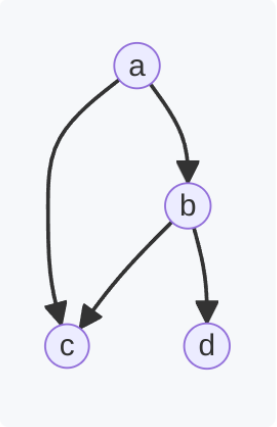
Злоумышленник пытается вывести из строя сервер, посылая случайные запросы каждую миллисекунду. При этом гарантируется, что случайный запрос не может совпасть с запросами типа 1 и типа 2 из-за их сложной структуры. В то же время пользователь сервера отправил 9 запросов обоих типов с интервалом в 15 миллисекунд в следующем порядке: {1 1 2 1 1 2 1 1 2}. И злоумышленник, и пользователь начали отправлять свои запросы одновременно в момент времени $t_0 = 0$.

Для защиты сервера администратор подключил защиту от DDoS-атак. Защита работает следующим образом: все принятые запросы становятся в единую бесконечно большую очередь на обработку. Каждый входящий запрос берётся из очереди и анализируется в течение некоторого времени и либо блокируется, либо пропускается дальше к вычислительному серверу. Обработка первых 30 запросов осуществляется за 2 миллисекунды на каждый запрос для определения факта атаки, и все эти запросы будут пропущены в очередь сервера независимо от типа. Обработка каждого следующего запроса осуществляется за 1 миллисекунду, при этом тип запроса не важен, но запросы от злоумышленника не пропускаются. При этом защита от DDoS-атак срабатывает не всегда и все-таки пропускает в очередь сервера каждый пятидесятый запрос злоумышленника, при этом 49 предыдущих его запросов блокируются. Защита от DDoS-атак анализирует запросы параллельно с работой сервера.

Определите, через сколько миллисекунд будет обработан последний запрос пользователя. Временем на передачу данных между узлами сети следует пренебречь (то есть если кто-то отправил запрос, он тут же в начале миллисекунды будет принят сервером, поставлен в очередь и при необходимости будет начата его обработка). Если на анализ поступает несколько пакетов в течение одной миллисекунды, в очередь сначала попадает пакет злоумышленника.

Ответ: 194

6. Операционные системы (3 балла) [Сколько вам пакетов?]



В операционных системах семейства GNU/Linux для установки ПО используется утилита, которая называется **менеджером пакетов** (или пакетным менеджером). Основная задача менеджера пакетов - определить, от каких пакетов зависит устанавливаемый пакет, и установить их вместе с ним. Такая операция производится рекурсивно для каждой зависимости, и в конце концов менеджер пакетов получает полный список зависимостей, который можно представить в виде графа. Такая операция называется **разрешением зависимостей**. Например, если для

установки пакета a нужны пакеты b и c, а для пакета b нужно установить пакеты c и d, то в итоге получится граф, изображённый на рисунке справа.

В данный момент сообществом разрабатывается экспериментальный пакетный менеджер. Рассмотрим принцип его работы.

Для работы менеджеру нужно иметь информацию о доступных пакетах в виде текстового файла определённого формата. Первая строчка описания каждого пакета начинается с ключевого слова `Package:`, после которого указывается название пакета. Вторая строчка описания содержит количество доступных версий и начинается с ключевого слова `Available-Versions:`. После этой строчки указывается информация о версиях. Каждая версия описывается тремя параметрами: номер версии (начинается с ключевого слова `Version:`), список необходимых зависимостей (начинается с ключевого слова `Depends:`) и размер скачиваемого файла (начинается с ключевого слова `Size:`). Пример информации о пакете с двумя версиями без зависимостей:

```
   Package: gcc-12-lib
Available-Versions: 2
Version: 12.3.0
Depends:
Size: 20068
Version: 12.2.8
Depends:
Size: 18880
```

При наличии зависимостей они описываются с использованием определённого формата. После ключевого слова `Depends:` через запятую указывается список пакетов, от которых зависит описываемый. Для каждой зависимости указывается её название, одна из операций сравнения и номер версии. Зависимости перечисляются через запятую. Пример информации о пакете с одной версией и двумя зависимостями:

```
   Package: libc6
Available-Versions: 1
Version: 2.35.5
Depends: libgcc-s1 >= 12.0.0, libcrypt1 >= 4.4.10
Size: 20068
```

Если в файле описано несколько пакетов, тогда описания разделяются пустой строкой.

Номер версии всегда состоит из трёх чисел, разделённых точкой. Первое число называется мажорной версией, второе - минорной версией, третье - патч-версией. Например, если версия равна 2.3.7, то мажорная версия равна 2, минорная - 3, патч-версия - 7.

Пакетный менеджер предусматривает следующие операции сравнения версий:

Операция	Описание
>>	Строго больше
>=	Больше или равна
==	Строго равна
~=	Больше или равна с сохранением мажорной и минорной версий
^=	Больше или равна с сохранением мажорной версии

При сравнении версий сначала сравниваются мажорные версии, затем минорные, затем патч-версии.

Граф зависимостей должен удовлетворять следующим правилам:

Зависимости каждого из пакетов в графе также должны быть в этом графе.

Изначально при установке пакета выбирается максимально возможная версия, подходящая под ограничения.

Если какая-то зависимость объявляется несколько раз (например, в разных пакетах), то версия пакета в итоговом варианте графа должна удовлетворять всем ограничениям.

Пакет нельзя установить, если какой-то из его зависимостей не существует либо не существует подходящей под ограничения версии зависимости. В таком случае нужно понижать версию проблемной зависимости и пробовать строить граф до тех пор, пока это не получится сделать. Если понижение версии зависимости не помогло, нужно понижать версию самого пакета, в случае необходимости продвигаясь по дереву вверх и выполняя понижение версии для всех пакетов на пути поочерёдно.

Необходимо построить граф зависимостей, соответствующий сформулированным требования, для пакета `zsh`.

Вам даётся доступ к поисковой строке, которая по названию пакета находит его служебную информацию в формате, описанном выше. Также Вы можете скачать текстовый файл https://olymp.itmo.ru/files/2024-01/5a59/4bcb/350bec6f-9d79-bf1d3a552c5a.txt с описанием всех пакетов. Определите, сколько пакетов будет установлено и каков общий размер скачиваемых файлов. В ответе введите два числа через пробел.

Ответ: 8 8309339

7. Технологии программирования (2 балла) [MindCraft]

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

В новой популярной игре «MindCraft» игрок управляет множеством фабрик по производству ресурсов. А конечная цель игры — производить определенные артефакты, для каждого из которых нужно фиксированное количество единиц определенных ресурсов. Всего есть n ресурсов, и i-я фабрика производит только i-й ресурс. Игрок может запускать и останавливать любые фабрики в любой момент времени. **Изначально все фабрики запущены.**

Также в игре есть особенный ресурс — *энергия*. Энергия нужна, чтобы фабрики работали: если энергии не хватает, фабрика не произведет соответствующий ресурс. Изначально у игрока есть A единиц энергии.

Каждый ход, именно в таком порядке:

игрок получает E единиц энергии;

игрок может запустить или остановить какие-то фабрики (можно ничего не менять);

запущенные фабрики последовательно выполняют работу: сначала первая, затем вторая, и т.д., и в самом конце — n-я;

выполняя работу, каждая запущенная в данный момент фабрика тратит энергию и производит ресурс: i-я фабрика тратит за ход ровно e_i энергии и производит x_i единиц i-го ресурса;

если какой-то фабрике не хватило энергии (текущее количество энергии у игрока строго меньше e_i), она в этот ход не работает.

В конце хода игрок может попробовать произвести какие-то артефакты. Всего есть m возможных артефактов, и для каждого из них известен его *рецепт* — какие ресурсы и в каком количестве нужны для его производства. При успешном производстве артефакта количество единиц каждого ресурса уменьшается на необходимую для производства величину. Если какого-то ресурса не хватает для производства, оно считается проваленным, и ресурсы не расходуются.

Просимулируйте действия игрока, и на каждый запрос производства артефакта сообщите, успешно ли он выполнен, и если нет — каких ресурсов не хватает.

Формат входных данных

Решите задачу для t уровней игры. В первой строке ввода дано целое число t — количество уровней, для которых требуется решить задачу ($1 \le t \le 40$). Далее следуют t описаний уровней.

Первая строка описания уровня содержит три целых числа A, E и n — изначальное количество энергии, сколько энергии добавляется каждый ход, и сколько типов ресурсов (и фабрик) есть в игре ($0 \le A, E \le 10^5$; $1 \le n \le 50$). В i-й из следующих n строк через пробел даны два целых числа e_i и x_i — расход энергии и количество единиц производимого за ход ресурса для i-й фабрики ($1 \le e_i, x_i \le 10^4$).

В следующей строке дано целое число m — количество артефактов в игре ($1 \le m \le 50$). В i-й из следующих строк дан рецепт i-го артефакта — n целых чисел $r_{i,j}$, записанных через пробел, означающих количество ресурса номер j, необходимое для производства этого артефакта ($0 \le r_{i,j} \le 10^5$).

В следующей строке дано целое число q — количество действий игрока ($1 \le q \le 1000$). В следующих q строках даны описания действий игрока. Действия даны в хронологическом порядке, i-е действие задается в формате «s_i ON f_i» или «s_i OFF f_i», если описывает запуск или остановку фабрики с номером f_i на ходу номер s_i, и в формате «s_i CREATE a_i», если описывает запрос на производство артефакта с номером a_i в конце хода s_i ($1 \le s_i \le 1000$; $1 \le f_i \le n$; $1 \le a_i \le m$).

Формат выходных данных

Для каждого запроса на производство артефакта выведите в отдельной строке «OK» (без кавычек), если артефакт был успешно произведен, иначе — выведите «FAIL: missing a», где a — **минимальный** номер ресурса, количества которого не хватает для его производства.

Пример

Стандартный ввод	Стандартный вывод
3 10 6 3 2 3 3 4 1 1 5 2 4 2 1 1 1 2 2 2 1 1 1 1 1 1 5 2 CREATE 1 3 CREATE 1 4 OFF 2 5 OFF 1 6 CREATE 5 10 3 4 1 6 3 1 8 2	OK FAIL: missing 3 OK FAIL: missing 4 FAIL: missing 3 OK FAIL: missing 2 OK

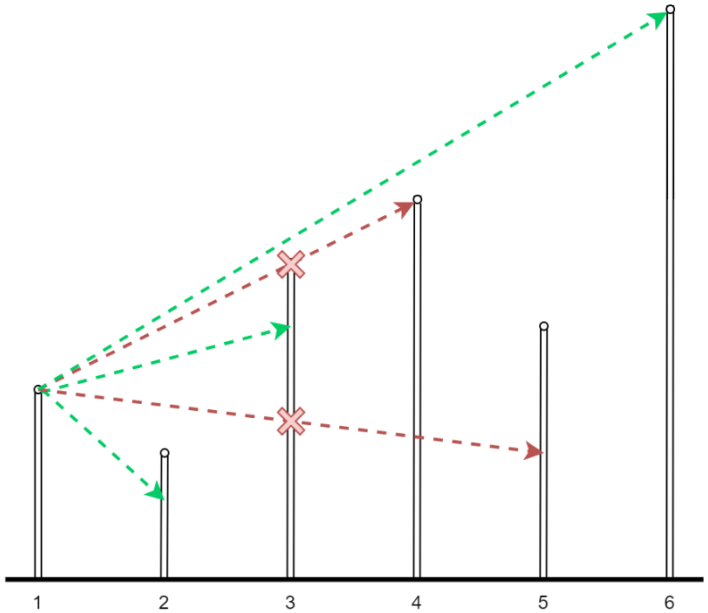
Стандартный ввод	Стандартный вывод
3 2 3 2 2 2 1 2 1 3 2 2 0 1 1 6 2 CREATE 1 3 OFF 1 3 OFF 3 8 CREATE 2 9 OFF 2 12 CREATE 3 1 15 3 4 6 8 2 6 3 2 3 2 1 3 7 4 4 3 CREATE 2 4 OFF 1 11 ON 1 12 CREATE 2	

8. Технологии программирования (4 балла) [Видеоконференция]

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

В Байтландии на одной из улиц в ряд стоят n небоскребов. Небоскреб номер i расположен в x_i метрах от начала улицы и имеет высоту y_i метров. Для простоты будем представлять i-й небоскреб как вертикальный отрезок длины y_i с нижним из двух концов на уровне земли.

На крыше первого небоскреба планируют разместить видовую площадку. Чтобы площадка действительно была видовой, с нее должно быть видно как можно больше других небоскребов на главной улице. Мы будем считать, что небоскреб i *виден* из точки (x_1, y_1), если существует отрезок с одним из концов в точке (x_1, y_1) и вторым концом на i-м небоскребе, не пересекающий ни один другой небоскреб (см. рисунок).



С крыши первого небоскреба видны: второй, третий и шестой.

Правительство города готово снести произвольное количество небоскребов, чтобы обеспечить наилучший вид с видовой площадки. Определите, какое максимальное количество небоскребов может быть видно с крыши первого, если разрешается снести произвольное их количество.

Формат входных данных

Решите задачу для t различных улиц Байтландии. В первой строке ввода дано целое число t — количество улиц, для которых надо решить задачу ($1 \le t \le 100$). Далее следуют t описаний улиц.

В первой строке описания улицы дано целое число n — количество небоскребов ($2 \le n \le 1000$). В i-й из следующих n строк через пробел даны целые числа x_i и y_i — расстояние от начала улицы до i-го небоскреба и его высота, соответственно ($0 \le x_i \le 10^4$; $1 \le y_i \le 10^4$). Гарантируется, что все x_i различны и что небоскребы перечислены в порядке возрастания x_i.

Гарантируется, что сумма n по всем t улицам не превосходит 1000.

Формат выходных данных

Выведите по очереди ответы на задачу для каждой из t улиц. В первой строке ответа выведите целое число k — максимальное количество небоскребов, которые может быть видно с видовой площадки. Во второй строке перечислите через пробел k различных целых чисел от 2 до n — номера этих небоскребов.

Обратите внимание, что требуется вывести именно номера тех небоскребов, которые будут видны с площадки, а не номера тех небоскребов, которые следует снести.

Пример

Стандартный ввод	Стандартный вывод
3 4 1 1 2 3 3 2 4 4 4 1 1 2 4 3 9 4 16 6 1 1 2 2 3 5 4 2 10 5 15 8	2 3 4 3 2 3 4 3 4 5 6

Задания для 9 и 10 класса

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

1. Кодирование информации. Системы счисления (1 балл) [XYZ_2024]

Дано выражение:

$$XYZ_9 + ZXY_{12} + YZX_{14} = 2024_{10}$$

В данном выражении X, Y и Z – допустимые различные цифры указанных систем счисления. Определите значения переменных Y и Z, если X = 5. В ответе укажите через пробел 2 целых положительных числа: сначала значение Y и затем значение Z.

Ответ: 6 2

Решение:

Так как Y и Z – допустимые цифры девятеричной системы счисления, то $0 \le Y, Z \le 8$

Переведём все числа данного выражения в десятичную систему:

$$XYZ_9 = 5 * 9 * 9 + Y * 9 + Z = 405 + 9Y + Z$$
$$ZXY_{12} = Z * 12 * 12 + 12 * 5 + Y = 144Z + 60 + Y$$
$$YZX_{14} = Y * 14 * 14 + Z * 14 + 5 = 196Y + 14Z + 5$$

Просуммируем слагаемые левой части:

$$405 + 9Y + Z + 144Z + 60 + Y + 196Y + 14Z + 5 = 206Y + 159Z + 470$$

Приравняем левую и правую часть, упростим:

$$206Y + 159Z + 470 = 2024$$
$$206Y + 159Z = 1554$$

Теперь необходимо более точно определить границы допустимых значений для Y и Z.

Во-первых,

$$1554 = 206Y + 159Z \ge 159(Y + Z)$$

Значит, $(Y + Z) \le 9$.

Во-вторых,

$$1554 = 206Y + 159Z \le 206(Y + Z)$$

Значит, $(Y + Z) \ge 8$.

В-третьих, так как число 1554 чётное, то либо оба слагаемых слева нечётные, либо оба чётные. Так как число 206Y не может быть нечётным вне зависимости от значения Y, то оба слагаемых чётные. Число 159Z чётное только при чётных значениях Z. Значит, возможные значения для числа 159Z – это 0, 318, 636, 954 и 1272. Соответствующие этим значениям значения для 206Y – 1554, 1236, 918, 600, 282. Заметим, что число 206Y делится нацело на 206. Из перечисленных 5 чисел таким свойством обладает только число 1236. Значит, Y=1236/206=6, Z=2. Ответ: 6 2.

Также возможно и решение задачи программированием, например, так:

```
from itertools import product
for y, z in product(range(9), repeat=2):
    num = int(f'5{y}{z}', 9) + int(f'{z}5{y}', 12) + int(f'{y}{z}5', 14)
    if y != z and num == 2024:
        print(y, z)
```

2. Кодирование информации. Объем информации (1 балл) [Цифровой диктофон]

Друг Пети, Павел, пытается сконструировать цифровой диктофон и просит Петю написать прошивку для кодирования и сохранения в памяти оцифрованного аудиосигнала. Петя решил, что будет записывать данные без сжатия и оцифровывать аудиосигнал с частотой дискретизации 88200 Hz, выбрав такую глубину кодирования, чтобы в каждый отсчет времени сохранялось одно из возможных 65536 значений сигнала (для записи значения сигнала в каждый отсчет времени Петя использует минимальное, одинаковое для всех возможных значений количество бит). Поскольку Петя предполагает использовать пару микрофонов, Павел решил записывать звук двухканальным, сохраняя оцифрованный аудиосигнал с указанными параметрами независимо для каждого канала. Опытный Вася обратил внимание Пети на две возможности для уменьшения памяти. Во-первых, можно уменьшить частоту дискретизации в два раза, а во-вторых, записывать с выбранной глубиной кодирования только один канал, а для второго канала записывать для каждого отсчета времени только разность значения сигнала со значением в первом канале, считая, что для этого хватит 1024 возможных значений (для записи разности сигналов в каждый отсчет времени предлагается также использовать минимальное, одинаковое для всех возможных значений разности количество бит). Петя принял оба предложения Васи и обнаружил, что для аудиосигнала длительностью t секунд удалось сэкономить больше 20 МБайт памяти. Определите минимальное **целое** значение t, при котором это возможно. В ответе укажите целое число.

Примечания:

1. При записи оцифрованного сигнала в память не записывается никакая дополнительная информация.
2. 1 МБайт=2^{20} байт.

Ответ: 101

Решение:

Запишем формулу для определения информационного объема в первоначальном формате записи:

$2 * t * 88200 * \log_2(65536) = t * 88200 * 2 * 16$ бит.

После изменения формата, частота дискретизации составит 88200/2=44100 Hz, а глубина кодирования для второго канала станет равна $\log_2(1024) = 10$ бит. Следовательно, формула для определения информационного объема будет: $t * 44100 * 16 + t * 44100 * 10 = t * 44100 * 26$ бит.

Определим значение t, при котором экономия памяти составила бы ровно 20 МБайт:

$t * 88200 * 2 * 16 - t * 44100 * 26 = 12 * 1024 * 1024 * 8$

$t = 20 * 1024 * 1024 * 8 / (88200 * 2 * 16 – 44100 * 26) = 100{,}11$ секунд

Следовательно, если бы длительность аудиосигнала была 100 секунд, разность составила бы меньше 20 МБайт, значит ответ – 101 секунда.

3. Основы логики (1 балл)

[Четыре импликации]

Два набора значений переменных A, B и C называют не эквивалентными, если значение хотя бы одной переменной различается.

Сколько существует не эквивалентных друг другу наборов значений переменных, при которых равенство выполняется при любом значении D:

$$(((D \to (A \wedge B \wedge C)) \to (A \wedge B \vee C)) \to (A \vee B \wedge C)) \to (A \vee B \vee C) = 1$$

В ответ укажите одно целое число.

Решение:

В первую очередь составим таблицу истинности для используемых выражений:

A	B	C	A∧B∧C	A∧B∨C	A∨B∧C	A∨B∨C
0	0	0	FALSE	FALSE	FALSE	FALSE
0	0	1	FALSE	TRUE	FALSE	TRUE
0	1	0	FALSE	FALSE	FALSE	TRUE
0	1	1	FALSE	TRUE	TRUE	TRUE
1	0	0	FALSE	FALSE	TRUE	TRUE
1	0	1	FALSE	TRUE	TRUE	TRUE
1	1	0	FALSE	TRUE	TRUE	TRUE
1	1	1	TRUE	TRUE	TRUE	TRUE

Рассмотрим таблицу истинности для импликации:

A	B	A->B
0	0	1
0	1	1
1	0	0
1	1	1

Заметим, что при B=1 результат импликации не зависит от A и равен 1. То есть, равенство $(\ldots) \to (A \vee B \vee C) = 1$ всегда верно при $(A \vee B \vee C) = 1$. Значит, 7 из 8 наборов значений A, B и C удовлетворяют условию. Проверим оставшийся набор. При A=0, B=0, C=0 выражение примет вид: $(((D \to 0) \to 0) \to 0) \to 0$. Рассмотрим значения этого выражения при D=0 и D=1

D	$D \to 0$	$(D \to 0) \to 0$	$((D \to 0) \to 0) \to 0$	$(((D \to 0) \to 0) \to 0) \to 0$
0	1	0	1	0
1	0	1	0	1

Таким образом, результат меняется в зависимости от D, что противоречит условию. То есть, 8-й набор не подходит, и ответ – 7.

4. Кодирование информации. Формальные исполнители (2 балла)

[6 букв]

Есть исходная строка S, состоящая из 6 символов. Результирующая строка R изначально пустая и формируется следующим образом: N раз выполняются следующие два шага:

3. Дописать в конец строки R строку S.
4. Циклически сдвинуть строку S на один символ влево.

Например, для строки S=’abcdef’ и N=4 получится строка R=’abcdefbcdefacdefabdefabc’.

Для некоторой строки S получили результирующую строку R для $N=10^{10}$.

Известны некоторые символы в строке R (нумерация символов строки начинается с 1 слева направо):

Номер символа	Значение
10^8+2	a
10^8+4	c
10^8+8	b
10^8+16	d
10^8+3	f
10^8+5	e

Определите и введите в ответ строку S, для которой это возможно.

Если вариантов таких строк несколько, введите любую подходящую.

Если такой строки не существует, введите в ответ NULL.

Ответ: cedabf

Решение:

Обратим внимание, что, поскольку в исходной строке 6 символов, возможно сделать последовательно 6 циклических сдвигов, после чего опять получится исходная строка. Следовательно, в результирующей строке будут повторяться одинаковые последовательности из 36 символов.

Построим такую последовательность (вручную или с помощью простой программы), взяв за основу строку «123456»:

123456234561345612456123561234612345

Возьмем первый номер из таблицы: 10^8+2 и поделим на 36 с остатком. Остаток будет равен 30. Следовательно, это 30-ый символ в этой строке – символ «4». Следовательно, четвертый символ в исходной строке, символ “a”.

Аналогично поступим с остальными номерами символов и получим следующую таблицу:

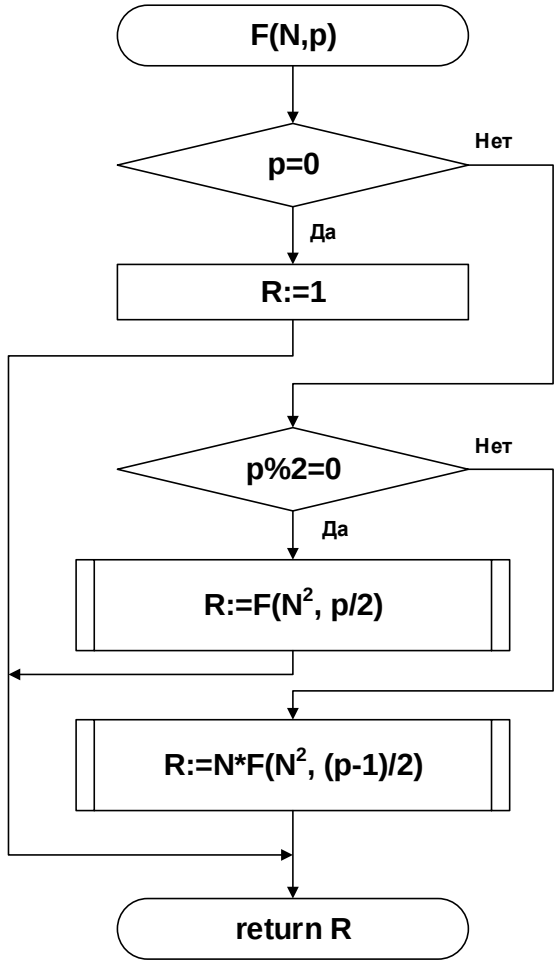
Номер символа в результирующей строке	Значение	Номер символа в исходной строке
10^8+2	a	4
10^8+4	c	1
10^8+8	b	5
10^8+16	d	3
10^8+3	f	6
10^8+5	e	2

Обратим внимание, что мы определили все 6 символов исходной строки. Запишем их в правильном порядке и получим ответ: cedabf.

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (2 балла)

[Рекурсия]

Дана блоксхема алгоритма, реализованного в виде рекурсивно вызываемой функции:



Найдите такую пару целых положительных чисел N и p (известно, что p>1), чтобы вызов F(N,p) вернул число 387420489. Если таких пар существует несколько, найдите ту, у которой максимальное значение N. В ответе укажите через пробел сначала значение N и затем значение p. Если такой пары не существует, укажите в ответе NULL.

Ответ: 19683 2

Решение:

Проанализировав алгоритм, представленный в виде блок-схемы, можно понять, что он реализует возведение числа N в степень p. Следовательно, нужно подобрать такие N и p, чтобы $N^p = 387420489$.

Для этого разложим данное в условие число на сомножители, например, с помощью такого программного решения:

```
ans=[]
d=2
while Z>1:
    if Z%d==0:
        ans.append(d)
        Z//=d
    else:
        d+=1
print(ans)
```

Результатом будет список: [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3], то есть 18 чисел 3. Значит $387420489 = 3^{18}$.

Но в условии сказано, что необходимо найти пару с максимальным значением N.

Заметим, что 18 = 9*2. Тогда $3^{18} = (3^9)^2 = 19\,683^2$. И это пара с максимальным значением N, поскольку значение p – минимально возможное (по условию p>1). Следовательно, ответ: 19683 2.

6. Телекоммуникационные технологии (3 балла). [Обрыв канала]

В ОС семейства GNU/Linux существует возможность объединения сетевых интерфейсов (сетевых карт) в группы (*bonding*). Ее используют для балансировки нагрузки при передаче и обеспечения отказоустойчивости.

В результате объединения в системе создается виртуальный сетевой интерфейс `bond`. С точки зрения остальных слоев сетевого стека этот интерфейс – обычная сетевая карта, однако при передаче данных через нее реальная передача данных осуществляется через тот или иной физический сетевой интерфейс по выбору ядра операционной системы. Выбор физического интерфейса зависит от режима *bonding*.

Существует 7 режимов *bonding*. Один из них – `balance-xor`. Когда физические интерфейсы объединяются в этом режиме, для выбора физического интерфейса, через который следует передать кадр канального уровня, используются

значения адресов канального уровня (MAC-адресов), содержащиеся в заголовке кадра. Расчет ведется по следующей формуле:

(MAC_отправителя XOR MAC_получателя) % число_физических_интерфейсов = индекс_физического_интерфейса

Здесь XOR – побитовая операция, а % - операция получения остатка от деления. Значения индекса физического интерфейса для передачи начинаются с нуля.

В случае недоступности части физических интерфейсов передача идёт только по активным интерфейсам (то есть изменяется число физических интерфейсов), а индексы самих интерфейсов пересчитываются, сохраняя изначальный порядок в конфигурации и нумерацию с нуля.

Рассмотрим систему с настроенным виртуальным сетевым интерфейсом bond в режиме balance-xor, который включает в себя 4 физических сетевых интерфейса с индексами от 0 до 3. Этой системе поступает несколько кадров для передачи. MAC-адрес отправителя равен 24:01:C7:A3:8B:7E, а MAC-адреса получателей указаны ниже.

```
00:24:1E:6D:FC:7D
44:45:53:36:A0:88
78:84:3C:94:06:A3
00:22:4C:FA:BE:C9
44:45:53:DC:E6:7A
68:76:4F:9A:99:FF
E0:0C:7F:53:C5:09
44:45:53:7C:70:65
BC:6E:64:3C:AD:EF
A4:5C:27:35:CE:7C
44:45:53:75:DA:1F
9C:5C:F9:74:17:50
00:09:BF:10:AD:FD
44:45:53:74:2A:F4
00:EB:2D:38:D0:F2
00:27:09:00:AC:1B
44:45:53:DC:53:CE
68:76:4F:18:EC:5C
64:B5:C6:58:97:DB
44:45:53:60:96:FE
00:1E:45:9C:9C:FF
E8:DA:20:CF:BB:CB
44:45:53:18:37:68
4C:21:D0:35:31:22
00:1F:C5:A3:8B:43
44:45:53:0B:25:65
00:21:9E:91:53:6B
00:26:59:B4:AE:9A
44:45:53:76:95:14
30:17:C8:8A:84:15
```

После передачи определённого количества кадров произошёл обрыв сети, из-за чего одновременно стали недоступны два физических интерфейса из четырёх, и оставшиеся кадры передавались по одному из двух активных интерфейсов. Известно, что по интерфейсу с исходным индексом 0 было передано 7 кадров, по интерфейсу с исходным индексом 1 – 12 кадров, по интерфейсу с исходным индексом 2 – 6 кадров, по интерфейсу с исходным индексом 3 – 5 кадров. Определите, какие из интерфейсов стали недоступны и сколько кадров было передано до обрыва канала. В ответе укажите три числа через пробел: номера интерфейсов (0, 1, 2 или 3) в порядке возрастания и количество переданных кадров. Если есть несколько вариантов того, какие интерфейсы могли стать недоступны, выберите любой вариант. Если есть несколько вариантов того, сколько кадров могло быть передано до обрыва, выберите максимальное число.

Ответ: 2 3 25

Решение:

Для решения задачи нам нужно посчитать побитовое исключающее ИЛИ для 30 пар 48-битных чисел и потом взять остаток от деления на количество активных интерфейсов. Можно заметить, что активных физических интерфейсов в любой момент времени либо 4, либо 2. Это позволяет нам посчитать исключающее ИЛИ только для последних 2 бит чисел пары и потом брать остаток от деления. Вычислим выражение из условия для каждой пары:

Адрес	Индекс интерфейса
00:24:1E:6D:FC:7D	3
44:45:53:36:A0:88	2
78:84:3C:94:06:A3	1

00:22:4C:FA:BE:C9	3
44:45:53:DC:E6:7A	0
68:76:4F:9A:99:FF	1
E0:0C:7F:53:C5:09	3
44:45:53:7C:70:65	3
BC:6E:64:3C:AD:EF	1
A4:5C:27:35:CE:7C	2
44:45:53:75:DA:1F	1
9C:5C:F9:74:17:50	2
00:09:BF:10:AD:FD	3
44:45:53:74:2A:F4	2
00:EB:2D:38:D0:F2	0
00:27:09:00:AC:1B	1
44:45:53:DC:53:CE	0
68:76:4F:18:EC:5C	2
64:B5:C6:58:97:DB	1
44:45:53:60:96:FE	0
00:1E:45:9C:9C:FF	1
E8:DA:20:CF:BB:CB	1
44:45:53:18:37:68	2
4C:21:D0:35:31:22	0
00:1F:C5:A3:8B:43	1
44:45:53:0B:25:65	3
00:21:9E:91:53:6B	1
00:26:59:B4:AE:9A	0
44:45:53:76:95:14	2
30:17:C8:8A:84:15	3

Дальше для рассмотрения возьмём вариант 1.

Нетрудно заметить, что в случае исправности всех интерфейсов в течение всего времени работы получилось бы следующее распределение кадров по интерфейсам:

Индекс	Количество (теор.)	Количество (факт.)
0	6	7
1	10	12
2	7	6
3	7	5

Видно, что интерфейсы с индексами 0 и 1 обработали больше кадров, а с индексами 2 и 3 - меньше, соответственно, первые два числа в ответе - 2 и 3.

Осталось понять, после какого кадра произошёл обрыв, для этого нужно посмотреть, когда мы передадим 1 лишний кадр через интерфейс 0 и 2 лишних кадра через интерфейс 1. Начнём пересчитывать номера интерфейсов ещё раз, но с конца:

Адрес	Индекс исходного интерфейса до обрыва	Индекс исходного интерфейса после обрыва	Не на тот интерфейс?
…	…	…	…
4C:21:D0:35:31:22	0	0	Нет
00:1F:C5:A3:8B:43	1	1	Нет
44:45:53:0B:25:65	3	1	Да
00:21:9E:91:53:6B	1	1	Нет
00:26:59:B4:AE:9A	0	0	Нет
44:45:53:76:95:14	2	0	Да
30:17:C8:8A:84:15	3	1	Да

Видно, что до обрыва должно быть передано максимум 25 кадров, чтобы получилась описанная в условии ситуация.

7. Технологии обработки информации в электронных таблицах, технологии сортировки и фильтрации данных (2 балла)

[Остаться самим собой]

Дана таблица в режиме отображения формул:

	A	B	C	D	E	F	G
1		2	3	4	5	6	
2	2	=IF(MOD(A1; POW(B$1; $A2))=0; DIVIDE($A$1; POW(B$1; $A2)); MOD($A$1; POW(B$1; $A2)))					
3	3						
4	4						
5	5						
6	6						
7	7						
8	8						

Формулу из ячейки B2 скопировали во все ячейки диапазона B2:F8. После чего оказалось, что число из ячейки A1 встречается в диапазоне B2:F8 ровно 4 раза. Определите минимальное и максимальное возможное число в ячейке A1, при котором это могло произойти. В ответ запишите сначала меньшее из значений, а затем большее из них.

Примечание: таблица соответствия имён используемых функций.

	Google Sheets	**Excel**	**LibreOffice**
Условное вычисление	IF	ЕСЛИ	IF
Возведение в степень	POW	СТЕПЕНЬ	POWER
Остаток от деления	MOD	ОСТАТ	MOD
Частное от целочисленного деления	DIVIDE	ЧАСТНОЕ	QUOTIENT

Ответ: 65536 78124

Решение:

Обозначим как X величину POW(B$1; $A2). Формула из ячейки B2 записывает в ячейку либо остаток от деления числа из ячейки A1 на X, если число из ячейки A1 делится не на X и результат деления в обратном случае. Следовательно, число из ячейки A1 (для краткости далее будем называть его просто A1) может попасть в одну из ячеек из диапазона B2:F8 двумя способами: либо если A1 делится на X и при этом A1/X=A1, т.е. X = 1, что не выполняется в данной таблице. Либо A1 не делится на X и тогда в ячейке будет значение A1 mod X. Данная величина будет равна A1 только в том случае, если A1 меньше X. То есть, A1 встретится в диапазоне B2:F8 четыре раза, если ровно 4 возможных значения POW(B$1; $A2) будут больше A1. Для каждого числа из диапазона B2:F8 определим значение X:

	A	B	C	D	E	F
1		2	3	4	5	6
2	2	4	9	16	25	36
3	3	8	27	64	125	216
4	4	16	81	256	625	1296
5	5	32	243	1024	3125	7776
6	6	64	729	4096	15625	46656
7	7	128	2187	16384	78125	279936
8	8	256	6561	65536	390625	1679616

Если A1 будет больше либо равно 78125, то только 3 числа в таблице будут больше A1. Значит, максимально возможное значение – 78124. Если A1 будет 65535 или меньше, то 5 чисел таблицы будут больше A1. Значит, A1 должно быть хотя бы 65536. Значит, ответ на данную задачу 65536 78124.

8. Технологии программирования (3 балла)

[Окна]

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

На экране монитора размером $w \times h$ расположены n прямоугольных окон нового текстового редактора. Размер экрана измеряется в символах, то есть текстовая строка длины w символов занимает всю ширину экрана от левого края до правого, а всего на экране могут поместиться друг под другом ровно h таких строк. Занумеруем строки экрана от 1 до h, а столбцы — от 1 до w, и будем обозначать координатами (i, j) позицию символа на пересечении i-й строки и j-го столбца.

Каждое из n окон содержит только текстовое поле, которое по ширине и высоте помещает в себе целое число символов. Окна могут быть разного размера и расположены так, что каждая из $w \times h$ позиций символов на экране принадлежит текстовому полю ровно одного окна. Иными словами, окна не могут накладываться, но могут касаться границами, и в совокупности покрывают всю площадь экрана.

Изначально все текстовые поля в окнах пустые, то есть не содержат никакой текст. Команды вывода строки на экран задаются тройками вида (r, c, s), означающими, что надо поставить курсор на c-ю слева позицию в r-ю сверху строку

монитора и, начиная с этой позиции, напечатать строку s. Символы строки печатаются по очереди слева направо, каждый следующий символ занимает следующую позицию в той же строке и записывается в ней вместо того, что стояло на этой позиции ранее. Так происходит, пока курсор не дойдет до правой границы текущего окна, после чего поведение курсора зависит от *режима*, в котором работает то окно, в котором он находится.

Окна могут работать в трех режимах. При достижении курсором правой границы окна:

В режиме «clip» вывод останавливается, и оставшаяся часть строки просто не выводится.

В режиме «wrap» курсор переносится на первую (ближайшую к левой границе окна) позицию **этого же окна** в следующей строке, после чего вывод продолжается. Если же текущая строка является последней строкой в текущем окне, вывод останавливается;

В режиме «overflow», если сейчас курсор находится в позиции экрана (r, c), он переносится в позицию $(r, c + 1)$, то есть в следующую позицию на экране, оставаясь в той же строке и игнорируя правую границу окна; после чего вывод продолжается. Если же текущая позиция является последней позицией на экране, то есть $c = w$, курсор переносится на первую позицию на экране в следующей строке $(r + 1,1)$. Если и строка была последняя, то есть курсор достиг правой-нижней позиции на экране ($r = h$ и $c = w$), вывод останавливается.

Для каждого окна известно, в каком режиме оно изначально работает. Обработайте команды вывода строк и команды изменения режимов окон и выведите состояние экрана после обработки всех команд.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода дано единственное целое число t — количество наборов входных данных в тесте ($1 \le t \le 50$).

Каждый набор входных данных начинается со строки, в которой даны три целых числа n, w и h — количество окон на экране и размеры экрана, соответственно ($1 \le w, h; 1 \le n \le w \cdot h \le 2000$).

Следующие n строк набора входных данных описывают окна; i-я строка описывает i-е окно и содержит через пробел четыре целых числа $r_{i,1}$, $c_{i,1}$, $r_{i,2}$ и $c_{i,2}$ — номер строки и столбца левого-верхнего угла окна и номер строки и столбца правого-нижнего угла окна ($1 \le r_{i,1} \le r_{i,2} \le h;\ 1 \le c_{i,1} \le c_{i,2} \le w$), а также строку mode_i, равную «clip», «wrap» или «overflow» — изначальный режим, в котором работает i-е окно. Гарантируется, что весь экран покрыт окнами и что каждая позиция на экране принадлежит ровно одному окну.

Затем в отдельной строке следует целое число q — количество команд, которые вам предстоит обработать ($1 \le q \le 2000$).

В i-й из следующих q строк дано описание i-й команды в формате

«print r_i c_i s_i», если это команда вывода строки s_i, начиная с позиции (r_i, c_i) ($1 \le r_i \le h; 1 \le c_i \le w; 1 \le |s_i|$);

«mode t_i m_i», если это команда изменения режима окна номер t_i на m_i ($1 \le t_i \le n$; $m_i \in$ «clip», «wrap», «overflow»).

Гарантируется, что все s_i состоят из маленьких букв латинского алфавита (символы от ‘a’ до ‘z’), и что сумма длин s_i по всем командам в рамках одного набора входных данных не превосходит $10\,000$.

Формат выходных данных

Для каждого набора входных данных выведите состояние экрана после обработки всех q команд. Состояние экрана — это h строк по w символов, каждый из которых равен букве на соответствующей позиции экрана, либо ‘.’, если соответствующая позиция пустая и не содержит символ.

Пример

Стандартный ввод	**Стандартный вывод**
2 2 2 2 1 1 2 1 clip 1 2 2 2 overflow 4 print 1 1 abcd print 1 2 xxxx mode 1 wrap print 1 2 yyyy 4 3 3 1 1 1 1 overflow 1 2 1 3 clip 2 1 3 1 wrap 2 2 3 3 wrap 8 print 1 1 abcd mode 2 overflow print 1 2 efgh mode 3 overflow	ay y. aef ijy zt.

Стандартный ввод	Стандартный вывод
print 2 1 ijkl print 2 3 x mode 4 overflow print 2 3 yzt	

Замечание

Для удобства восприятия наборы входных данных и соответствующий им вывод в примерах в условии отделяются друг от друга пустыми строками. В реальных тестах этих **пустых строк нет**!

Решение:

Это задача на реализацию. Требовалось внимательно прочитать условие и реализовать то, что в нем просили, добавив минимальные необходимые оптимизации.

Заведем

- text[r][c] – символ, стоящий на позиции (r,c);
- id[r][c] – номер окна, покрывающего позицию с координатами (r,c);
- mode[i] – режим, в котором работает окно номер i;
- left[i] и bottom[i] – номер левого столбца и нижней строки i-го окна, соответственно.

Изначально необходимо просто заполнить таблицу id в соответствии с данными в условии, после чего приступить к обработке запросов. Для каждого запроса будем поддерживать текущее положе- ние курсора (r, c) и очередной символ строки, который надо вывести. Обозначим также за $\mathrm{id}_{\mathrm{cur}}$ номер текущего окна, то есть id[r][c].

Тогда сначала присвоим в text[r][c] текущий символ, а затем переместим курсор.

- Если $c < w$ и id[r][$c+1$] = $\mathrm{id}_{\mathrm{cur}}$, то есть мы еще не дошли до правой границы окна, просто увеличим c на 1. Иначе, посмотрим на mode[$\mathrm{id}_{\mathrm{cur}}$] - режим текущего окна.
- Если режим равен «clip», сделаем break из цикла, отвечающего за вывод, чтобы его остано- вить.
- Если режим равен «wrap», перенесем курсор в $r \leftarrow r+1$ и $c \leftarrow \mathrm{left}[\mathrm{id}_{\mathrm{cur}}]$. Следует отдельно проверить, что если $r = \mathrm{bottom}[\mathrm{id}_{\mathrm{cur}}]$, то тоже надо сделать break.
- Если же окно работает в режиме «overflow», то либо курсор переместится в $(r,c+1)$, либо в
- $(r+1,1)$, либо просто следует сделать break, если достигнут правый нижний угол экрана.

Каждый запрос в таком случае обрабатывается за $O(|s_i|)$, и в сумме все решение работает за $O(wh + \sum |s_i|)$. Если же каждый раз искать номер текущего окна перебором, не храня id[r][c], решение с большой вероятностью не будет проходить по времени.

В другом варианте этой задачи режимам «clip» и «overflow» соответствовали «fill» и «break», а «clip» был заменен на «repeat». В режиме «repeat» достаточно было просто перемещать курсор на $(r, \mathrm{left}[\mathrm{id}_{\mathrm{cur}}])$.

9. Технологии программирования (3 балла)

[Скрещивание]

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

Вы — владелец зоопарка, в котором сейчас обитают n редких экзотических видов змей, i-й из которых имеет *опасность* a_i.

Держать животных в неволе вам не очень хочется, но и выпускать опасных змей в общее пространство тоже, разумеется, нельзя. Для исправления этой проблемы вы решили скрестить некоторые виды, от чего их экзотичность меньше не станет, а вот опасность может уменьшиться. Для одного скрещивания вы можете выбрать два вида змей под номерами i и j, и скрестить их в новый вид, имеющий опасность $a^*_{i,j} = a_i \oplus a_j$, где за $\oplus$ обозначена операция побитового исключающего «ИЛИ» (также обозначается как xor или ^).

Чтобы не получать слишком похожие виды, каждый из имеющихся n видов может участвовать только в одном скрещивании. Также невозможно скрестить два вида, хотя бы один из которых уже является результатов скрещивания каких-то из изначальных n видов. Иными словами, скрещивать можно только исходные виды, и только по парам.

В конце вы получаете новый набор видов, в который войдут исходные виды, не поучаствовавшие в скрещиваниях, а также все результаты скрещиваний. То есть исходные виды, которые были скрещены с какими-то еще, в этот набор не войдут (опять же, потому что нет смысла отводить в зоопарке отдельное место под несколько близких видов — посетители все равно не увидят разницу).

Определите, какие пары видов змей следует скрестить, чтобы максимум из опасностей полученного набора видов был как можно меньше.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода дано единственное целое число t — количество наборов входных данных в тесте ($1 \le t \le 50$). Далее следуют сами наборы входных данных.

В первой строке набора входных дано целое число n — количество видов экзотических змей в наличии изначально ($1 \le n \le 50\,000$). Гарантируется, что сумма n по всем наборам входных данных не превосходит 10^5.

Во второй строке перечислены n целых чисел a_i — значения опасности этих видов ($0 \le a_i \le 10^9$).

Формат выходных данных

Для каждого набора входных данных выведите в отдельной строке единственное целое число — минимальное возможное значение максимальной опасности, которое можно получить такими скрещиваниями.

Пример

Стандартный ввод	**Стандартный вывод**
4 3 1 2 3 7 1 4 9 16 25 36 49 6 0 1 2 5 6 7 10 5 7 14 10 12 2 1 13 3 11	1 21 4 8

Замечание

В первом примере выгодно скрестить 2 и 3, и получить исходный вид с $a_1 = 1$ и новый с $a_{2,3} = 1$.

Во втором примере скрещиваются 16 с 25 (получается $a_{4,5} = 9$) и 36 с 49 (получается $a_{6,7} = 21$).

В третьем примере скрещиваются 2 с 6 и 5 с 7.

Решение:

Пойдем с конца: пусть ответом на задачу является число x. Посмотрим на его битовую запись и найдем в ней первую (старшую) единицу lead(x). Пусть она стоит на позиции y с конца, то есть соответствует 2^y при разложении x в сумму степеней двойки. Тогда заметим, что любой a_i, у которого lead(a_i) > y, то есть старшая единица стоит раньше, должен быть поставлен в пару с a_j, у которого все биты старше y совпадают с a_i, чтобы их xor давал в старших битах 0.

Таким образом, чтобы в ответе lead был равен y, все a_i с lead(a_i) > y должны разбиваться на пары с одинаковым $\text{head}_{y+1}(a_i) = a_i >> (y+1)$. Здесь за >> обозначен битовый сдвиг вправо, то есть операция, отбрасывающая последние биты числа. В данном случае $a_i >> (y+1)$ отбрасывает все биты с нулевого по y-й включительно.

Алгоритм получается следующий: переберем y от 29 до 0 (можно вместо перебора сделать двоичный поиск, чтобы немного увеличить эффективность), сгруппируем все a_i по их head_y, то есть по $30 - y$ старшим битам, и проверим, что все группы, у которых head_y ненулевой, содержат четное число элементов. Если это так, то можно добиться того, чтобы во всех $30 - y$ старших битах у каждого числа в ответе стояли нули, просто сгруппировав исходные числа по парам внутри групп.

Как только мы нашли такой y, для которого хотя бы одна из групп оказалась нечетного размера, понятно, что в ответе в соответствующем бите будет стоять 1. При этом по всем head_{y-1} группы все еще четного размера, поэтому числа надо разбивать на пары внутри таких групп. Тогда сгруппируем a_i по их head_{y-1} и внутри каждой группы выделим те, у которых следующий бит равен 0 и те, у которых следующий бит равен 1.

После этого для каждой группы решим задачу независимо.

Если количество тех, у которых следующий бит равен 1, четно, то их можно внутри этой группы разбить на пары так, чтобы все результирующие числа имели в следующем бите 0, и в таком случае они все будут меньше ответа.

Иначе можно разбить на пары все, кроме одного. А еще одно число либо оставить как есть, и тогда ответ для этой группы просто будет равен минимальному из чисел с единицей в сле- дующем бите, либо сопоставить ему в пару число из той же группы, но с нулем в следующем бите.

В таком случае достаточно найти $\min_{a_i \in group0,\ a_j \in group1} a_i \oplus a_j$,

где group0 и group1 - элементы текущей группы с нулем или единицей в следующем бите, соответственно.

Поиск такого минимума является классической задачей, которая решается с помощью битового бора: будем воспринимать каждое число как строку из 30 нулей и единиц, добавим все элементы одной подгруппы в такой бор, а для каждого элемента второй подгруппы будем искать ему пару, минимизирующую их xor, стараясь каждый раз в боре идти по совпадающему биту.

Теперь, когда в каждой группе получено минимально возможное максимальное число, ответом будет максимум из ответов для всех групп. Общее время работы такого решения - O(n log A), где A - ограничение сверху на значения a_i.

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

1. Кодирование информации. Системы счисления (1 балл)

[XYZ_2024]

Дано выражение:

$$XYZ_9 + ZXY_{12} + YZX_{14} = 2024_{10}$$

В данном выражении X, Y и Z – допустимые различные цифры указанных систем счисления. Определите значения переменных Y и Z, если X = 5. В ответе укажите через пробел 2 целых положительных числа: сначала значение Y и затем значение Z.

Ответ: 6 2

Решение:

Так как Y и Z – допустимые цифры девятеричной системы счисления, то $0 \le Y, Z \le 8$

Переведём все числа данного выражения в десятичную систему:

$$XYZ_9 = 5 * 9 * 9 + Y * 9 + Z = 405 + 9Y + Z$$
$$ZXY_{12} = Z * 12 * 12 + 12 * 5 + Y = 144Z + 60 + Y$$
$$YZX_{14} = Y * 14 * 14 + Z * 14 + 5 = 196Y + 14Z + 5$$

Просуммируем слагаемые левой части:

$$405 + 9Y + Z + 144Z + 60 + Y + 196Y + 14Z + 5 = 206Y + 159Z + 470$$

Приравняем левую и правую часть, упростим:

$$206Y + 159Z + 470 = 2024$$
$$206Y + 159Z = 1554$$

Теперь необходимо более точно определить границы допустимых значений для Y и Z.

Во-первых,

$$1554 = 206Y + 159Z \ge 159(Y + Z)$$

Значит, $(Y + Z) \le 9$.

Во-вторых,

$$1554 = 206Y + 159Z \le 206(Y + Z)$$

Значит, $(Y + Z) \ge 8$.

В-третьих, так как число 1554 чётное, то либо оба слагаемых слева нечётные, либо оба чётные. Так как число 206Y не может быть нечётным вне зависимости от значения Y, то оба слагаемых чётные. Число 159Z чётное только при чётных значениях Z. Значит, возможные значения для числа 159Z – это 0, 318, 636, 954 и 1272. Соответствующие этим значениям значения для 206Y – 1554, 1236, 918, 600, 282. Заметим, что число 206Y делится нацело на 206. Из перечисленных 5 чисел таким свойством обладает только число 1236. Значит, Y=1236/206=6, Z=2. Ответ: 6 2.

Также возможно и решение задачи программированием, например, так:

```
from itertools import product
for y, z in product(range(9), repeat=2):
    num = int(f'5{y}{z}', 9) + int(f'{z}5{y}', 12) + int(f'{y}{z}5', 14)
    if y != z and num == 2024:
        print(y, z)
```

2. Кодирование информации. Объем информации (2 балла) [Canis aureus]

Для кодирования цветов часто используется RGB-палитра. В этом случае цвет каждого пикселя изображения кодируется с помощью трех отдельных числовых значений, называемыми цветовыми каналами (красный (R), зеленый (G) и синий (B)). Числовое значение по каждому цветовому каналу в общем случае может быть в диапазоне от 0 до 255. При сохранении каждого числового значения в памяти для него отводится минимальное, одинаковое для всех значений количество бит. Илья решил, что может хранить изображения используя меньше памяти.

Во-первых, он применил к каждому пикселю изображения следующий алгоритм сжатия:

1. Если значения всех цветовых каналов имеют одинаковую чётность - цвет не меняется.
2. Если только одно из значений нечётное - оно уменьшается на 1.
3. Если только одно из значений чётное - оно увеличивается на 1.

Во-вторых, он решил, что после сжатия будет отводить минимальное, одинаковое для всех значений количество бит не значению каждого цветового канала пикселя, а коду, который присваивается уникальной комбинации значений всех цветовых каналов каждого пикселя. Уникальной является комбинация, отличающаяся от любой другой значением хотя бы одного цветового канала.

Теперь Илья хочет оценить эффективность своей идеи. Определите, на сколько байт уменьшилось количество памяти, необходимое для хранения изображения размером 1920 на 1080 пикселей. Если необходимое количество памяти увеличилось – запишите в ответ отрицательное число. Например, если раньше требовалось 100 байт, а после преобразований потребуется 98 байт, то ответ будет равен 2, а если потребуется 102 байта, то ответ будет равен -2.

Ответ: 518400

Решение:

Определим, сколько бит необходимо для хранения одного пикселя до преобразований.

В задаче рассматривается кодирование значений минимально возможным одинаковым количеством бит,

следовательно, количество бит определяется по формуле Хартли: количество бит $= log_2$(количество значений), с округлением до ближайшего большего целого числа. Для хранения каждого цветового канала понадобится $log_2(255 - 0 + 1) = 8$ бит. Так как каналов 3, потребуется 24 бита на каждый пиксель.

Определим, сколько бит необходимо для хранения одного пикселя после преобразований.

В первую очередь вычислим, сколько возможных цветов (то есть, комбинаций из трёх значений цветовых каналов) было до сжатия. Так как каждый цветовой канал может иметь 256 различных значений, и значений цветовых каналов не зависят друг от друга, то возможных комбинаций $256 * 256 * 256 = 2^{24}$.

Теперь выясним, как изменится количество возможных цветов при сжатии. Численное значение каждого цветового канала можно представить в виде $2n + k$, где n – целое число от 0 до 127 включительно, а k равно 0 для чётного числа и 1 для нечётного. Значит, все возможные цвета (комбинации значений трёх цветовых каналов) можно разбить на 8 групп:

1. $(2n + 0, 2n + 0, 2n + 0)$
2. $(2n + 0, 2n + 0, 2n + 1)$
3. $(2n + 0, 2n + 1, 2n + 0)$
4. $(2n + 0, 2n + 1, 2n + 1)$
5. $(2n + 1, 2n + 0, 2n + 0)$
6. $(2n + 1, 2n + 0, 2n + 1)$
7. $(2n + 1, 2n + 1, 2n + 0)$
8. $(2n + 1, 2n + 1, 2n + 1)$

Применим к данным группам преобразования согласно алгоритму. Группы 1 и 8 не изменятся, так как все числа имеют одинаковую чётность. Группы 2, 3, 5 изменят значение на $(2n + 0, 2n + 0, 2n + 0)$, так как имеют только одно нечётное значение. То есть, перейдут в группу 1. Группы 4, 6, 7 изменят значение на $(2n + 1, 2n + 1, 2n + 1)$, так как имеют только одно чётное значение. То есть, перейдут в группу 8. Таким образом, из 8 групп мы получим только 2, то есть различных комбинаций станет в 4 раза меньше. До преобразования их было 2^{24}, т.е. теперь их будет 2^{22}. Для каждого пикселя мы храним информацию о его цвете, различных цветов у нас 2^{22}, значит потребуется по 22 бита на пиксель. То есть, на каждый пиксель требуется на 2 бита меньше.

Вычислим изменение необходимого количества памяти для всего изображения. Потребуется на $1920 * 1080 * 2$ бит меньше, т.е. на 4147200 бит = 518400 байт меньше.

3. Основы логики (1 балл)

[Четыре импликации]

Два набора значений переменных A, B и C называют не эквивалентными, если значение хотя бы одной переменной различается.

Сколько существует не эквивалентных друг другу наборов значений переменных, при которых равенство выполняется при любом значении D:

$$(((D \rightarrow (A \wedge B \wedge C)) \rightarrow (A \wedge \mathrm{B} \vee \mathrm{C})) \rightarrow (\mathrm{A} \vee \mathrm{B} \wedge \mathrm{C})) \rightarrow (A \vee \mathrm{B} \vee \mathrm{C}) = 1$$

В ответ укажите одно целое число.

Решение:

В первую очередь составим таблицу истинности для используемых выражений:

A	B	C	A∧B∧C	A∧B∨C	A∨B∧C	A∨B∨C
0	0	0	FALSE	FALSE	FALSE	FALSE
0	0	1	FALSE	TRUE	FALSE	TRUE
0	1	0	FALSE	FALSE	FALSE	TRUE
0	1	1	FALSE	TRUE	TRUE	TRUE
1	0	0	FALSE	FALSE	TRUE	TRUE
1	0	1	FALSE	TRUE	TRUE	TRUE
1	1	0	FALSE	TRUE	TRUE	TRUE
1	1	1	TRUE	TRUE	TRUE	TRUE

Рассмотрим таблицу истинности для импликации:

A	B	A->B
0	0	1
0	1	1
1	0	0
1	1	1

Заметим, что при B=1 результат импликации не зависит от A и равен 1. То есть, равенство $(\ldots) \rightarrow (A \vee \mathrm{B} \vee \mathrm{C}) = 1$ всегда верно при $(A \vee \mathrm{B} \vee \mathrm{C}) = 1$. Значит, 7 из 8 наборов значений A, B и C удовлетворяют условию. Проверим оставшийся набор. При A=0, B=0, C=0 выражение примет вид: $(((D \rightarrow 0) \rightarrow 0) \rightarrow 0) \rightarrow 0$. Рассмотрим значения этого выражения при D=0 и D=1

D	$D \rightarrow 0$	$(D \rightarrow 0) \rightarrow 0$	$((D \rightarrow 0) \rightarrow 0) \rightarrow 0$	$(((D \rightarrow 0) \rightarrow 0) \rightarrow 0) \rightarrow 0$
0	1	0	1	0

1	0	1	0	1

Таким образом, результат меняется в зависимости от D, что противоречит условию. То есть, 8-й набор не подходит, и ответ – 7.

4. Алгоритмизация и программирование. Формальные исполнители (2 балла)

[Бедная лошадка]

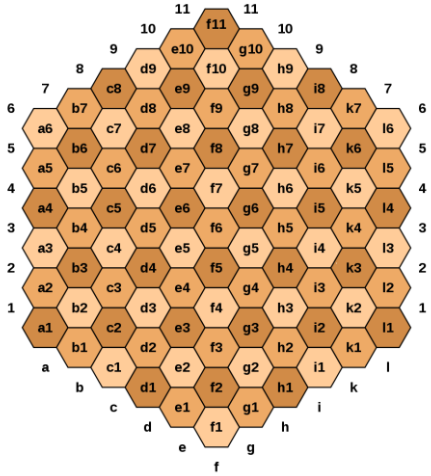
Недавно Ксюша узнала о существовании гексагональных шахмат. Больше всего её заинтересовали перемещения коней и слонов. Она задала набор дополнительных правил для перемещения коня:

1. Он может сделать сколько угодно корректных ходов.
2. Он не может наступать на клетки, находящиеся под боем у слонов (клетка находится под боем, если слон может попасть в неё за один корректный ход).
3. Он не может съедать слонов.
4. Слоны не могут двигаться.

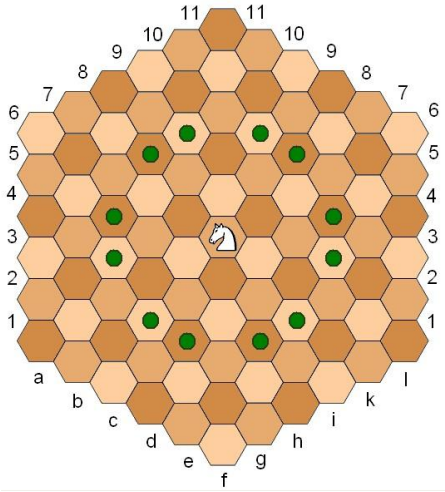
Ксюша решила определить, сколько есть клеток на поле, посещение которых конем не будет противоречить указанным условиям, но которые конь тем не менее не сможет посетить, если он будет начинать на клетке **e6**, а слоны будут стоять на клетках **d6**, **e9** и **i6**? В ответе напишите сначала количество клеток, а затем в возрастающем лексикографическом порядке через пробел эти клетки. Например, 3 a2 a3 b2. Если окажется, что таких клеток нет, напишите в ответ NULL.

Примечание:

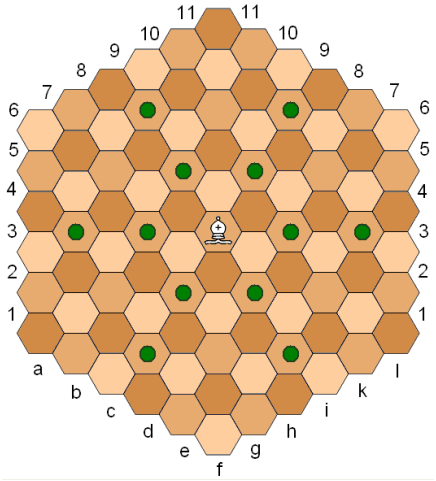
Нумерация клеток шахматной доски



Ход коня



Ход слона



Роль горизонталей выполняют косые линии полей, параллельные одной из невертикальных сторон доски, а роль диагоналей — линии полей одного цвета

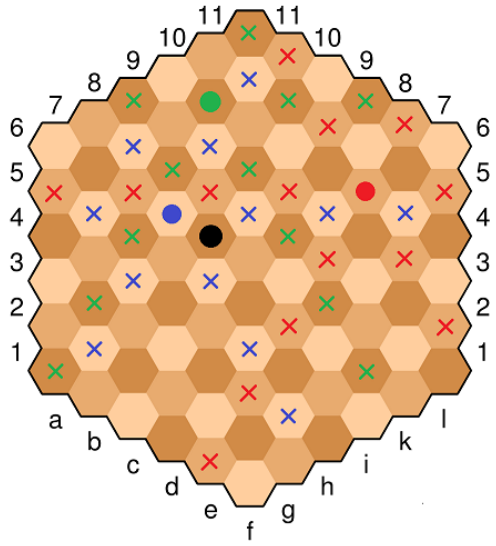
Конь ходит на два поля по вертикали или «горизонтали» и ещё на одно поле по другой вертикали или «горизонтали» (поворот на 120 градусов). Не обязательно, чтобы промежуточные клетки были свободны/достижимы, главное, чтобы были свободны начальная и конечная клетки.

Слон ходит на любое количество полей по диагоналям своего цвета.

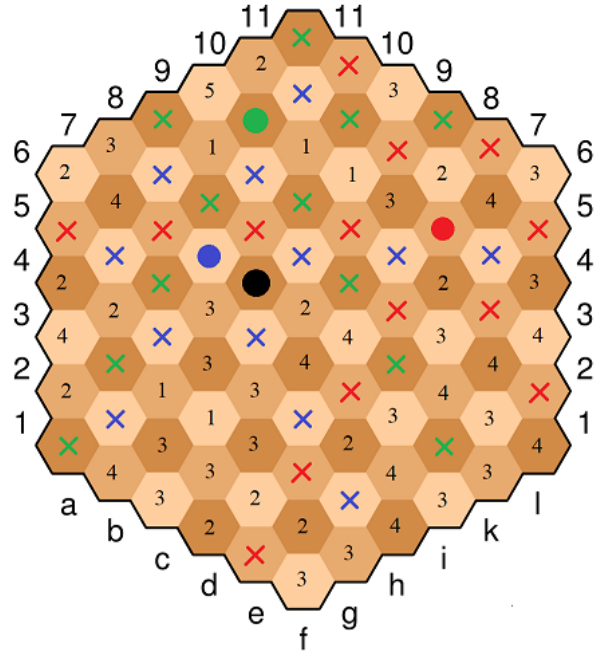
Ответ: NULL

Решение:

Для решения данной задачи необходимо проэмулировать возможные перемещения коня. Для начала обозначим стартовые позиции фигур. Конь обозначен чёрным кругом, слоны – синим, красным и зелёным кругами. Крестиками соответствующего цвета обозначим клетки, находящиеся под ударом у данного слона:



Теперь обозначим цифрой 1 все клетки, которые конь может достить из своего положения за 1 ход, цифрой 2 – те, что можно достичь за 1 ход из клеток, отмеченных цифрой 2, и так далее. Если клетка уже отмечена цифрой, менять её не будем.

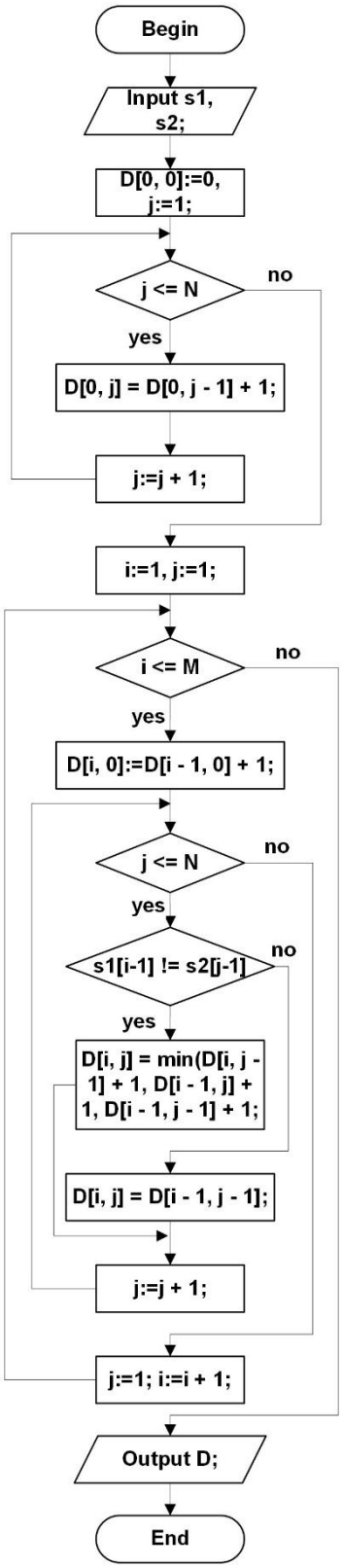


Как видно из рисунка, коню удалось посетить все клетки, разрешенные условиями, значит, ответ NULL.

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

[Два не очень разных слова]

Приведена блок-схема алгоритма:



В данной диаграмме: s1, s2 – строки из строчных букв английского алфавита; M – длина строки s1; N – длина строки s2; D – матрица размером M+1 на N+1. Строки матрицы нумеруются от 0, сверху вниз. Столбцы нумеруются от 0, слева направо. Запись D[i, j] означает ячейку на пересечении строки i и столбца j. Символы строк также нумеруются от 0.

В результате выполнения была получена следующая матрица:

0 1 2 3 4 5 6
1 1 2 2 3 4 5
2 2 2 3 2 3 4
3 3 3 3 3 2 3
4 4 4 4 3 3 2
5 5 5 5 4 4 3

6 6 6 6 5 5 4
7 7 7 7 6 6 5

Определите, какой могла быть строка s2, если строка s1=”**monocle**” (символы кавычек не являются частью строки). В случае, если символ строки s2 не удаётся определить однозначно, замените его символом *. Например, если в строке из пяти символов первые два определить не удалось, а остальные 3 – это символы **y, m, p,** то в качестве ответа необходимо указать **ymp.

Ответ: **mono

Решение:

Приведенный алгоритм рассчитывает, сколько действий вида «заменить символ, вставить символ, удалить символ» требуется для преобразования одной строки в другую. (Заметим, что число преобразований не зависит от порядка строк). Значение в D[i, j] соответствует числу действий, необходимых для преобразования подстроки из первых i символов одной строки в подстроку из первых j символов другой.

В первую очередь необходимо понять, по какому принципу заполняется матрица D. Для начала заполняются 0-я строка и 0-й столбец:

0 1 2 3 4 5 6
1 ? ? ? ? ? ?
2 ? ? ? ? ? ?
3 ? ? ? ? ? ?
4 ? ? ? ? ? ?
5 ? ? ? ? ? ?
6 ? ? ? ? ? ?
7 ? ? ? ? ? ?

После чего остальная таблица заполняется по следующему принципу: если (i-1)-й символ строки s1 и (j-1)-й символ строки s2 равны, то в ячейку D[i, j] записывается значение из ячейки D[i-1, j-1]. Именно на это свойство нам следует опираться при определении совпадения символов. Если символы не совпадают, то мы выбираем наименьший из 3 вариантов: либо заменить текущий символ (тогда нам понадобится D[i-1, j-1]+1 действий), либо удалить его (тогда нам понадобится D[i-1, j]+1 действий), либо добавить символ (тогда нам понадобится D[i, j-1]+1 действий). Обратите внимание, что если в ячейках D[i-1, j-1] и D[i, j] содержатся одинаковые значения, но при этом одно или оба значения D[i-1, j], D[i, j-1] равняются D[i, j]-1, то однозначно определить, были ли символы s1[i-1] и s2[j-1] нельзя.

Пользуясь данными знаниями, выделим в результирующей таблице все D[i, j], которые равны D[i-1, j-1]:

0 1 2 3 4 5 6
1 1 2 2 3 4 5
2 2 2 3 2 3 4
3 3 3 3 3 2 3
4 4 4 4 3 3 2
5 5 5 5 4 4 3
6 6 6 6 5 5 4
7 7 7 7 6 6 5

Отсечём те из них, «происхождение» которых нельзя определить однозначно (значение слева или сверху на единицу меньше):

0 1 2 3 4 5 6
1 1 2 2 3 4 5
2 2 2 3 2 3 4
3 3 3 3 3 2 3
4 4 4 4 3 3 2
5 5 5 5 4 4 3
6 6 6 6 5 5 4
7 7 7 7 6 6 5

Оставшиеся выделенными символы являются результатом равенства символов. Для удобства допишем слева от таблицы слово s1.

  0 1 2 3 4 5 6
m 1 1 2 2 3 4 5
o 2 2 2 3 2 3 4
n 3 3 3 3 3 2 3
o 4 4 4 4 3 3 2
c 5 5 5 5 4 4 3
l 6 6 6 6 5 5 4
e 7 7 7 7 6 6 5

Обозначим все неизвестные символы строки s2 звёздочками и будем заменять по мере выяснения однозначно определяемых символов. Так как число столбцов на 1 больше числа символов строки s2, то s2=******

Т.к. D[1, 3] написано в результате равенства s1[0] и s2[2], s2=**m***

Т.к. D[2, 4] написано в результате равенства s1[1] и s2[3], s2=**mo**

Т.к. D[3, 5] написано в результате равенства s1[2] и s2[4], s2=**mon*

Т.к. D[4, 6] написано в результате равенства s1[3] и s2[5], s2=**mono

Значение D[4, 4] также написано в результате равенства, но s2[3] мы уже установили из D[2, 4].

Оставшиеся символы однозначно установить невозможно, значит ответ **mono.

6. Телекоммуникационные технологии (1 балл)

[Локальное потепление]

Некоторый сервер с недостаточным охлаждением умеет обрабатывать запросы двух типов. Запрос типа А он обрабатывает за 8 секунд, а запрос типа Б – за 12 секунд. Однако, за каждую секунду, во время которой сервер обрабатывает запрос (вне зависимости от типа запроса), его температура увеличивается на 2 градуса. Если температура сервера достигнет 140 градусов, он необратимо выйдет из строя. Для того чтобы избежать этого, системный администратор решил после обработки каждого запроса (вне зависимости от типа) делать перерыв в X секунд. За каждую секунду перерыва температура сервера снижается на 0.5 градуса. Определите минимально возможное натуральное значение X, при котором сервер никогда не нагреется до 140 градусов, если запросы приходят в следующем порядке: А, А, Б, А, А, Б, А, А, Б и т.д. Температура сервера на момент начала работы – 50 градусов.

Ответ: 38

Решение:

Заметим, что порядок появления запросов состоит из повторяющихся последовательностей А, А, Б. Очевидно, что если спустя 3 таких запроса и 3 перерыва (после каждого из запросов) суммарное изменение температуры будет положительным, то рано или поздно процессор нагреется до 140 градусов и выйдет из строя. Составим выражение, описывающее изменение температуры за такой период: суммарная длительность процессов А, А, Б составит 8+8+12=28 секунд. То есть, за это время от нагреется на 28*2=56 градусов. А остынет – на 3*X*0.5=1.5X. Значит, изменение температуры составит 56-1.5X. Так как процессор не должен нагреваться, $56 - 1.5X <= 0$. То есть, $X \geq 37,(3)$. Наименьшее натуральное число, удовлетворяющее данному неравенству – 38.

7. Технологии обработки информации в электронных таблицах, технологии сортировки и фильтрации данных (2 балла)

[Остаться самим собой]

Дана таблица в режиме отображения формул:

	A	B	C	D	E	F	G
1		2	3	4	5	6	
2	2	=IF(MOD(A1; POW(B$1; $A2))=0; DIVIDE($A$1; POW(B$1; $A2)); MOD($A$1; POW(B$1; $A2)))					
3	3						
4	4						
5	5						
6	6						
7	7						
8	8						

Формулу из ячейки B2 скопировали во все ячейки диапазона B2:F8. После чего оказалось, что число из ячейки A1 встречается в диапазоне B2:F8 ровно 4 раза. Определите минимальное и максимальное возможное число в ячейке A1, при котором это могло произойти. В ответ запишите сначала меньшее из значений, а затем большее из них.

Примечание: таблица соответствия имён используемых функций.

	Google Sheets	**Excel**	**LibreOffice**
Условное вычисление	IF	ЕСЛИ	IF
Возведение в степень	POW	СТЕПЕНЬ	POWER
Остаток от деления	MOD	ОСТАТ	MOD
Частное от целочисленного деления	DIVIDE	ЧАСТНОЕ	QUOTIENT

Ответ: 65536 78124

Решение:

Обозначим как X величину POW(B$1; $A2). Формула из ячейки B2 записывает в ячейку либо остаток от деления числа из ячейки A1 на X, если число из ячейки A1 делится не на X и результат деления в обратном случае. Следовательно, число из ячейки A1 (для краткости далее будем называть его просто A1) может попасть в одну из ячеек из диапазона B2:F8 двумя способами: либо если A1 делится на X и при этом A1/X=A1, т.е. X = 1, что не выполняется в данной таблице. Либо A1 не делится на X и тогда в ячейке будет значение A1 mod X. Данная величина будет равна A1 только в том случае, если A1 меньше X. То есть, A1 встретится в диапазоне B2:F8 четыре раза, если ровно 4 возможных значения POW(B$1; $A2) будут больше A1. Для каждого числа из диапазона B2:F8 определим значение X:

	A	B	C	D	E	F
1		2	3	4	5	6
2	2	4	9	16	25	36
3	3	8	27	64	125	216
4	4	16	81	256	625	1296
5	5	32	243	1024	3125	7776
6	6	64	729	4096	15625	46656
7	7	128	2187	16384	78125	279936
8	8	256	6561	65536	390625	1679616

Если A1 будет больше либо равно 78125, то только 3 числа в таблице будут больше A1. Значит, максимально возможное значение – 78124. Если A1 будет 65535 или меньше, то 5 чисел таблицы будут больше A1. Значит, A1 должно быть хотя бы 65536. Значит, ответ на данную задачу 65536 78124.

8. Технологии программирования (3 балла)

[Окна]

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

На экране монитора размером $w \times h$ расположены n прямоугольных окон нового текстового редактора. Размер экрана измеряется в символах, то есть текстовая строка длины w символов занимает всю ширину экрана от левого края до правого, а всего на экране могут поместиться друг под другом ровно h таких строк. Занумеруем строки экрана от 1 до h, а столбцы — от 1 до w, и будем обозначать координатами (i, j) позицию символа на пересечении i-й строки и j-го столбца.

Каждое из n окон содержит только текстовое поле, которое по ширине и высоте помещает в себе целое число символов. Окна могут быть разного размера и расположены так, что каждая из $w \times h$ позиций символов на экране принадлежит текстовому полю ровно одного окна. Иными словами, окна не могут накладываться, но могут касаться границами, и в совокупности покрывают всю площадь экрана.

Изначально все текстовые поля в окнах пустые, то есть не содержат никакой текст. Команды вывода строки на экран задаются тройками вида (r, c, s), означающими, что надо поставить курсор на c-ю слева позицию в r-ю сверху строку монитора и, начиная с этой позиции, напечатать строку s. Символы строки печатаются по очереди слева направо, каждый следующий символ занимает следующую позицию в той же строке и записывается в ней вместо того, что стояло на этой позиции ранее. Так происходит, пока курсор не дойдет до правой границы текущего окна, после чего поведение курсора зависит от *режима*, в котором работает то окно, в котором он находится.

Окна могут работать в трех режимах. При достижении курсором правой границы окна:

В режиме «clip» вывод останавливается, и оставшаяся часть строки просто не выводится.

В режиме «wrap» курсор переносится на первую (ближайшую к левой границе окна) позицию **этого же окна** в следующей строке, после чего вывод продолжается. Если же текущая строка является последней строкой в текущем окне, вывод останавливается;

В режиме «overflow», если сейчас курсор находится в позиции экрана (r, c), он переносится в позицию $(r, c + 1)$, то есть в следующую позицию на экране, оставаясь в той же строке и игнорируя правую границу окна; после чего вывод продолжается. Если же текущая позиция является последней позицией на экране, то есть $c = w$, курсор переносится на первую позицию на экране в следующей строке $(r + 1,1)$. Если и строка была последняя, то есть курсор достиг правой-нижней позиции на экране ($r = h$ и $c = w$), вывод останавливается.

Для каждого окна известно, в каком режиме оно изначально работает. Обработайте команды вывода строк и команды изменения режимов окон и выведите состояние экрана после обработки всех команд.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода дано единственное целое число t — количество наборов входных данных в тесте ($1 \le t \le 50$).

Каждый набор входных данных начинается со строки, в которой даны три целых числа n, w и h — количество окон на экране и размеры экрана, соответственно ($1 \le w, h; 1 \le n \le w \cdot h \le 2000$).

Следующие n строк набора входных данных описывают окна; i-я строка описывает i-е окно и содержит через пробел четыре целых числа $r_{i,1}$, $c_{i,1}$, $r_{i,2}$ и $c_{i,2}$ — номер строки и столбца левого-верхнего угла окна и номер строки и столбца правого-нижнего угла окна ($1 \le r_{i,1} \le r_{i,2} \le h;\ 1 \le c_{i,1} \le c_{i,2} \le w$), а также строку mode_i, равную «clip», «wrap» или «overflow» — изначальный режим, в котором работает i-е окно. Гарантируется, что весь экран покрыт окнами и что каждая позиция на экране принадлежит ровно одному окну.

Затем в отдельной строке следует целое число q — количество команд, которые вам предстоит обработать ($1 \le q \le 2000$).

В i-й из следующих q строк дано описание i-й команды в формате

«print r_i c_i s_i», если это команда вывода строки s_i, начиная с позиции (r_i, c_i) ($1 \le r_i \le h; 1 \le c_i \le w; 1 \le |s_i|$);

«mode t_i m_i», если это команда изменения режима окна номер t_i на m_i ($1 \le t_i \le n$; $m_i \in$ «clip», «wrap», «overflow»).

Гарантируется, что все s_i состоят из маленьких букв латинского алфавита (символы от ‘a’ до ‘z’), и что сумма длин s_i по всем командам в рамках одного набора входных данных не превосходит 10 000.

Формат выходных данных

Для каждого набора входных данных выведите состояние экрана после обработки всех q команд. Состояние экрана — это h строк по w символов, каждый из которых равен букве на соответствующей позиции экрана, либо ‘.’, если соответствующая позиция пустая и не содержит символ.

Пример

Стандартный ввод	**Стандартный вывод**
2 2 2 2 1 1 2 1 clip 1 2 2 2 overflow 4 print 1 1 abcd print 1 2 xxxx mode 1 wrap print 1 2 yyyy 4 3 3 1 1 1 1 overflow 1 2 1 3 clip 2 1 3 1 wrap 2 2 3 3 wrap 8 print 1 1 abcd mode 2 overflow print 1 2 efgh mode 3 overflow print 2 1 ijkl print 2 3 x mode 4 overflow print 2 3 yzt	ay y. aef ijy zt.

Замечание

Для удобства восприятия наборы входных данных и соответствующий им вывод в примерах в условии отделяются друг от друга пустыми строками. В реальных тестах этих **пустых строк нет**!

Решение:

Это задача на реализацию. Требовалось внимательно прочитать условие и реализовать то, что в нем просили, добавив минимальные необходимые оптимизации.

Заведем

- text[r][c] - символ, стоящий на позиции (r, c);
- id[r][c] - номер окна, покрывающего позицию с координатами (r, c);
- mode[i] - режим, в котором работает окно номер i;
- left[i] и bottom[i] - номер левого столбца и нижней строки i-го окна, соответственно.

Изначально необходимо просто заполнить таблицу id в соответствии с данными в условии, после чего приступить к обработке запросов. Для каждого запроса будем поддерживать текущее положе- ние курсора (r, c) и очередной символ строки, который надо вывести. Обозначим также за $\mathrm{id}_{\mathrm{cur}}$ номер текущего окна, то есть id[r][c].

Тогда сначала присвоим в text[r][c] текущий символ, а затем переместим курсор.

- Если $c < w$ и id[r][$c + 1$] = $\mathrm{id}_{\mathrm{cur}}$, то есть мы еще не дошли до правой границы окна, просто увеличим c на 1. Иначе, посмотрим на mode[$\mathrm{id}_{\mathrm{cur}}$] - режим текущего окна.
- Если режим равен «clip», сделаем break из цикла, отвечающего за вывод, чтобы его остано- вить.
- Если режим равен «wrap», перенесем курсор в $r \leftarrow r + 1$ и $c \leftarrow$ left[$\mathrm{id}_{\mathrm{cur}}$]. Следует отдельно проверить, что если r = bottom[$\mathrm{id}_{\mathrm{cur}}$], то тоже надо сделать break.
- Если же окно работает в режиме «overflow», то либо курсор переместится в (r, $c + 1$), либо в
- ($r + 1$, 1), либо просто следует сделать break, если достигнут правый нижний угол экрана.

Каждый запрос в таком случае обрабатывается за $O(|s_i|)$, и в сумме все решение работает за $O(wh + |s_i|)$. Если же каждый раз искать номер текущего окна перебором, не храня id[r][c], решение с большой вероятностью не будет проходить по времени.

В другом варианте этой задачи режимам «clip» и «overflow» соответствовали «fill» и «break», а «clip» был заменен на «repeat». В режиме «repeat» достаточно было просто перемещать курсор на (r, left[id_{cur}]).

9. Технологии программирования (3 балла)

[Делители факториала]

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

Математика порой бывает довольно сложной. Однако, когда речь заходит о сложной математике, мало кто может сказать, что тема «факториалы» однозначно попадает в эту категорию. С задачей посчитать факториал целого числа, скорее всего, справится любой участник олимпиады.

Поэтому вам предстоит ответить на немного более сложные запросы. Всего запросов q и i-й запрос задается одним целым числом n_i. Для каждого запроса вам необходимо посчитать $\sigma_0(n_i!)$, то есть количество положительных делителей числа $n_i!$.

Поскольку реальное количество делителей числа $n!$ может быть слишком большим уже даже при $n \approx 100$, посчитайте ответ на каждый запрос по модулю $10^9 + 7$.

Формат входных данных

В первой строке ввода дано целое число q — количество запросов ($1 \le q \le 300$).

Во второй строке перечислены q целых чисел n_i — параметры запросов ($1 \le n_i \le 10^5$).

Формат выходных данных

Для каждого запроса посчитайте $\sigma_0(n_i!) \bmod (10^9 + 7)$ и выведите в ответ **остаток их суммы по модулю** 998 244 353.

Пример

Стандартный ввод	Стандартный вывод
10 1 2 3 4 5 6 7 10 100 1000	558634260

Замечание

В примере из условия независимые ответы для каждого n_i получаются такие:

$\sigma_0(1!) = 1$;

$\sigma_0(2!) = 2$;

$\sigma_0(3!) = 4$;

$\sigma_0(4!) = 8$;

$\sigma_0(5!) = 16$;

$\sigma_0(6!) = 30$;

$\sigma_0(7!) = 60$;

$\sigma_0(10!) = 270$;

$\sigma_0(100!) = 39\,001\,250\,856\,960\,000$, и по модулю $10^9 + 7$ это равно 583951250;

$\sigma_0(1000!) \bmod (10^9 + 7) = 972926972$, само значение мы приводить не будем, так как оно слишком большое.

Решение

Можно запустить линейное решето Эратосфена, которое позволяет для каждого числа x найти $\mathrm{lp}[x]$ – минимальное простое число в разложении x на простые.

Затем, вместо того, чтобы независимо отвечать на все запросы, просто предподсчитаем ответ для всех возможных n, и будем для ответа на запрос возвращать предподсчитанное значение. Для этого будем поддерживать $\deg[i]$ – степень вхождения i-го простого числа в $n!$.

Для начала запомним, что $\sigma_0(1!) = 1$, и затем будем перебирать n от 2 до 10^5, каждый раз пересчитывая значение ответа. Поскольку $n! = (n-1)! \cdot n$, то изменятся только степени вхождения простых, на которые делится n. Используя lp, можно за $O(\log n)$ разложить n на простые и для каждого p^{a_i} в его разложении поделить ответ на предыдущее значение $(\deg[i]+1)$, увеличить $\deg[i]$ на a_i и умножить ответ обратно на $(\deg[i]+1)$.

Модули $10^9 + 7$ и $10^9 + 9$ - простые, поэтому деление по ним можно осуществлять с помощью умножения на обратное, а для этого тоже можно воспользоваться двоичным возведением в степень, потому что по малой теореме Ферма $x^{M-2} = x^{-1}$.

Такое решение в сумме работает $O(A \log A)$ времени.

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

1. Кодирование информации. Системы счисления (1 балл) [Баланс единиц]

Сколько существует натуральных чисел X, меньших 10000 таких, что и запись числа X в двоичной системе счисления, и запись числа X в четверичной системе счисления содержат ровно 4 единицы? В ответе укажите целое число.

Ответ: 35

2. Кодирование информации. Количество информации. Элементы комбинаторики(2 балла) [Три монеты]

В мешочке лежат монеты разных номиналов – 1, 2 и 5. Монеты имеют одинаковый размер и вес. Монет номинала 1 - 10 штук, номинала 2 - 13 штуки, номинала 5 - 42 штуки. Из мешочка достали 3 монеты. Определите, сколько бит информации несёт в себе сообщение, что все монеты оказались различных номиналов. В ответе укажите целое число (при необходимости, округлить до ближайшего большего целого).

Ответ: 3

3. Кодирование информации. Количество информации. Кодирование текста (2 балла) [Птичья соцсеть]

В некоторой социальной сети публикация составляется из латинских строчных и заглавных букв, 12 различных символов пунктуации из набора { !, ?, ., @, -, :, /, *, (,), ;, , }, пробелов и 512 видов пиктограмм. Лимит суммы всех символов и пиктограмм в одной публикации – 141, при этом пиктограммы могут составлять не более трети сообщения.

Символы и пиктограммы кодируются раздельно, для кодирования каждого символа используется минимально возможное, одинаковое для всех символов количество бит и для кодирования каждой пиктограммы используется минимально возможное, одинаковое для всех пиктограмм количество бит. Для того, чтобы различать в закодированном виде символы и пиктограммы перед кодом символа всегда дописывается 0, а перед кодом пиктограммы – 1. Так как количества символов, соответствующие коду символа и коду пиктограммы известны, подобный приём позволяет обеспечить однозначность декодирования.

Сколько бит составляет максимальный информационный объём публикации? В ответе укажите целое число.

Ответ: 1222

4. Кодирование информации. Объем данных (3 балла) [Мгновенный повтор]

Вася разрабатывает для школьных киберспортивных соревнований систему мгновенного повтора, позволяющую записать последние несколько секунд экрана участника. Для соревнований предполагается использовать мониторы с разрешением 2560x1440 пикселей и стандартной палитрой RGB цветов (24 бита на пиксель), а чтобы ничего не упустить, Вася хочет записывать видео с частотой 60 кадров/секунду. Для того, чтобы уменьшить размер записи, Вася полностью сохраняет целиком только каждый k-й кадр, начиная с самого первого, а для последующих k-1 кадров он сохраняет только их отличие от предыдущего. То есть, для каждого из таких частично сохраненных кадров он запоминает набор информации об изменившихся пикселях. Для каждого пикселя записываются следующие четыре параметра: номер кадра после сохранённого целиком (число от 1 до k-1 включительно), вертикальная координата пикселя (число от 1 до 1440 включительно), горизонтальная координата пикселя (число от 1 до 2560 включительно) и новый цвет. Каждый из четырех параметров кодируется независимо, используется равномерное кодирование, то есть каждое значение записывается одинаковым, минимально возможным для всех значений этого параметра количеством бит последовательно, друг за другом.

Вася хочет подобрать такое k до 5000, чтобы каждая запись имела максимально возможную длительность и занимала не более 240 Мбайт памяти. Для упрощения своей задачи он считает, что каждый кадр отличается от предыдущего не более чем 1000 пикселями.

Определите, какое наибольшее целое число секунд удастся хранить Васе.

Ответ: 416

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

Определите, сколько существует различных комбинаций значений переменных A, B и C, которые могут являться решениями следующей системы уравнений:

$$\begin{cases} (A \to B) \to (B \to C) = A \wedge B \to B \wedge C \\ A \wedge B \to B \wedge C = A\ XOR\ B \to B\ XOR\ C \end{cases}$$

В ответ укажите через пробел сначала количество решений, а затем через пробел в лексикографическом порядке сами эти решения в формате #1#2#3, где #1 – значение переменной A (0 – ложь, 1 – истина), #2 – значение переменной B (0 – ложь, 1 – истина), #3 – значение переменной C (0 – ложь, 1 – истина). Если система не имеет решений ни при каких значениях переменных, в ответ укажите NULL.

Пример записи ответа: 2 000 111

Ответ: 4 000 001 101 111

6. Основы логики. Упрощение логического выражения (1 балл) [Нет или нет]

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

$$\overline{A \wedge \bar{B} \wedge C \wedge \bar{D} \wedge E \wedge \bar{F} \wedge G \wedge \bar{H}} \vee \overline{B \wedge \bar{C} \wedge D \wedge \bar{E} \wedge F \wedge \bar{G} \wedge H \wedge \bar{I}}$$

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

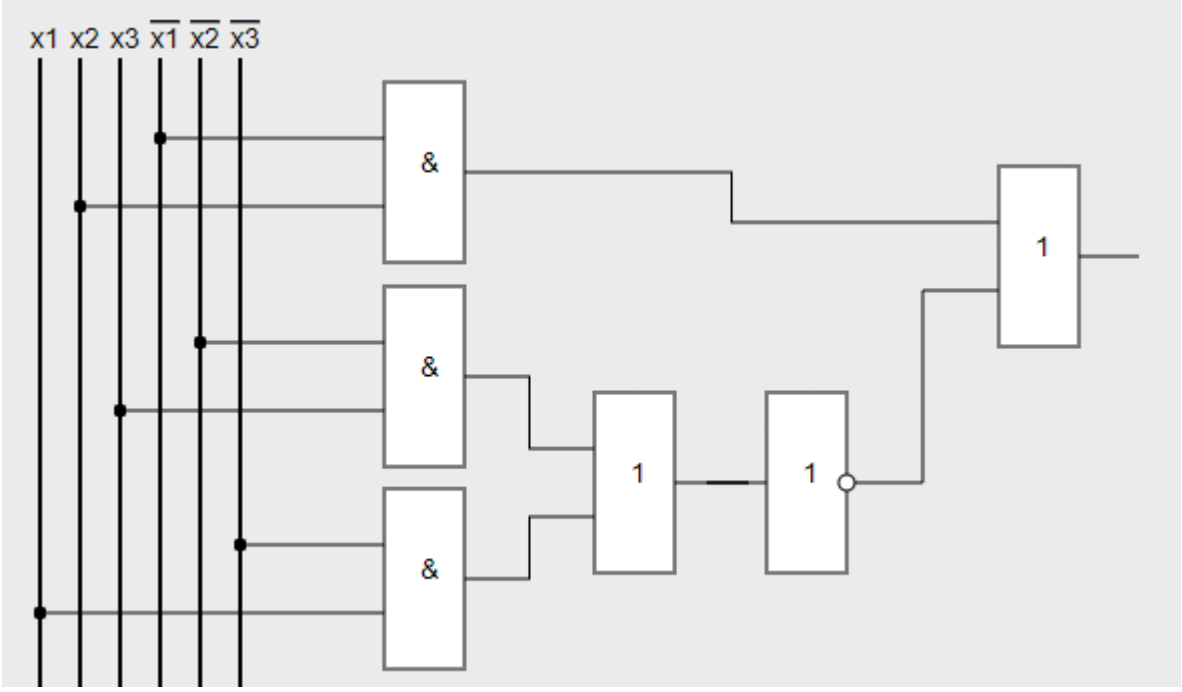
При однозначном ответе – истинный ответ обозначается как 1, а ложный как 0.

Пример записи ответа: A or B and not C

Ответ: 1

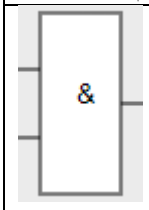
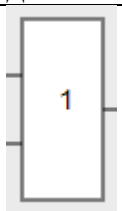
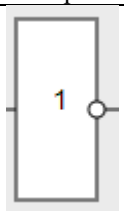
7. Основы логики. Синтез выражения по логической схеме (1 балл) [Логическая схема]

Дана схема логической функции F от трёх переменных:



Сколько существует различных наборов значений переменных x1, x2, x3 таких, что в результате будет получено значение 1 (истина)? В ответе укажите число.

Примечание. На схеме использованы следующие обозначения логических операторов:

Конъюнкция	Дизъюнкция	Инверсия
		

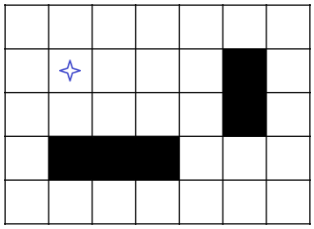
Ответ: 4

8. Алгоритмизация и программирование. Формальный исполнитель (3 балла) [Морской бой]

Настя разрабатывает генератор стартовых полей для игры в морской бой. Генератор выполняет итерации следующего алгоритма до тех пор, пока не сможет успешно разместить все корабли или исполнение не будет прервано самим алгоритмом:

1. Выбрать свободную клетку поля, которую не выбирали ранее. Алгоритм выбирает одну случайную клетку из множества тех, что ещё не были выбраны ранее и не заняты размещённым кораблём. До начала работы алгоритма все клетки поля входят в это множество.
2. Если в одной из восьми соседних клеток (соседними считаются клетки, соприкасающиеся с данной клеткой по стороне или углу) уже есть корабль – итерация считается неудачной и алгоритм переходит к следующей итерации. Если предыдущие 4 итерации также закончились неудачей - алгоритм прекращает свою работу с ошибкой.
3. Вычисляются 2 величины A и B: число свободных клеток поля справа от выбранной и число свободных клеток поля ниже выбранной соответственно (включая текущую клетку). Свободными считаются клетки поля, которые не соседствуют с размещёнными кораблями. Например, если в данной итерации выбрана клетка, отмеченная

звёздочкой, то величина A будет составлять 3, а величина B – 1.



4. Выбирается ориентация корабля: если предыдущий корабль расположили вертикально, новый корабль будет расположен горизонтально и наоборот.
5. Определяется максимальный размер корабля, который можно разместить в заданном направлении: если выбрано горизонтальное направление, длина корабля не должна превышать A, если выбрано вертикальное – не должна превышать B.
6. В выбранном направлении, начиная с текущей клетки, размещается самый большой из возможных кораблей. То есть, в случае горизонтального направления корабль размещается в текущей клетке и клетках справа от неё, в случае вертикального – в текущей клетке и клетках ниже неё. Например в ситуации на рис. 1, если выбрано горизонтально направление и необходимо разместить ещё один корабль длины 2 и один корабль длины 1, то будет размещён корабль длины 2, а если выбрано вертикальное – то корабль длины 2 разместить не удастся, и будет размещён корабль длины 1. А в ситуации на рис. 2 корабль длины 2 удастся разместить в любом из направлений.

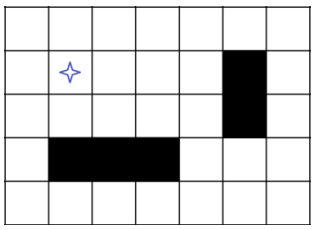


Рисунок 1

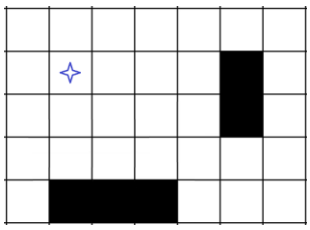


Рисунок 2

7. Итерация заканчивается

Согласно правилам игры необходимо разместить 4 корабля длиной в 1 клетку, 3 корабля длиной в 2 клетки, 2 корабля длиной 3 клетки и 1 корабль длиной 4 клетки. Первый корабль размещается горизонтально. Известно, что были последовательно выбраны следующие клетки (строки таблицы пронумерованы сверху вниз буквами A-J, столбцы пронумерованы слева направо числами 1-10):

B2
C4
D3
F9
G6
D6
G2
I4
J9
J2
C8
H8
B10

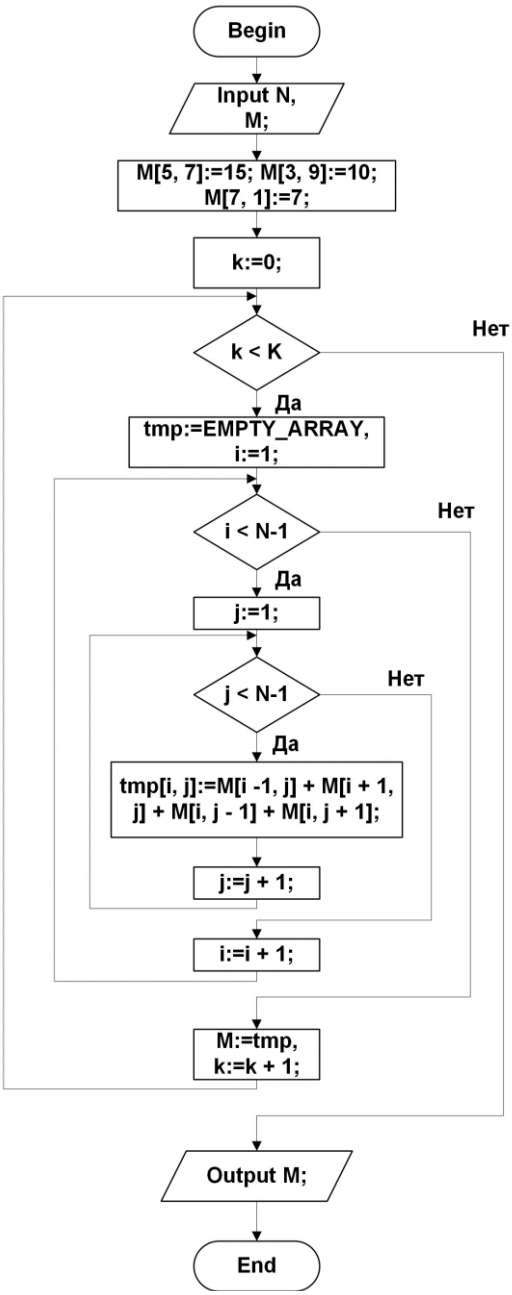
Определите клетки, для которых итерация прошла неудачно. В ответ запишите нужные клетки через пробел в порядке выбора их алгоритмом.

Пример записи ответа: B2 A9 C10

Ответ: C4 G2 C8

9. Алгоритмизация и программирование. Блок-схема, массивы, обратная задача (2 балла) [Сколько было итераций?]

Дана блок-схема алгоритма:

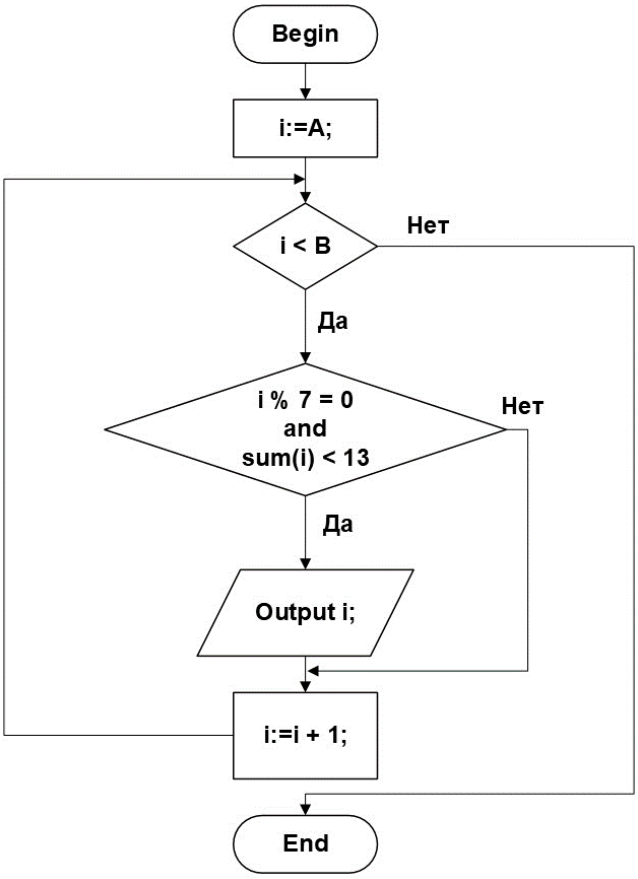


M – массив размера N на N элементов, изначально заполненный нолями. EMPTY_ARRAY, tmp также означают массивы N на N элементов, изначально заполненные нолями. Определите, каким могло быть минимальное возможное значение параметра K, если N = 10, а самое большое число, выведенное в результате работы алгоритма равно 10603890.

Ответ: 12

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

Дана блок-схема алгоритма:



Определите минимальное возможное B, если на экран было выведено 3950 чисел, а A = 100000. Функция sum(i) означает вычисление суммы цифр числа i.

Ответ: 2100001

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

1. Электронные таблицы. Адресация ячеек и вычисления (2 балла) [Что и по какому модулю]

В ячейки таблицы B1 и A2 записали некоторые формулы:

	A	B
1		=A$1+1
2	=POW(MOD(A1; XX); 2)	
3		

В формуле в ячейке A2 XX обозначает некоторое (необязательно двузначное) число. После чего формулу из ячейки B1 скопировали во все ячейки диапазона B1:M1, а формулу из ячейки A2 во все ячейки диапазона A2:M20. В ячейку A1 написали некоторое число, не кратное числу XX. В результате было обнаружено, что в диапазоне E2:E20 число, отличное от 1, содержится только в ячейке E2. Определите наименьшее возможное значение для XX и для него наименьшее возможное число в ячейке A1, если в ячейке E2 содержится число 900. В ответ запишите через пробел 2 числа – сначала XX, затем число из ячейки A1.

Примечание: таблица соответствия имён используемых фукнций.

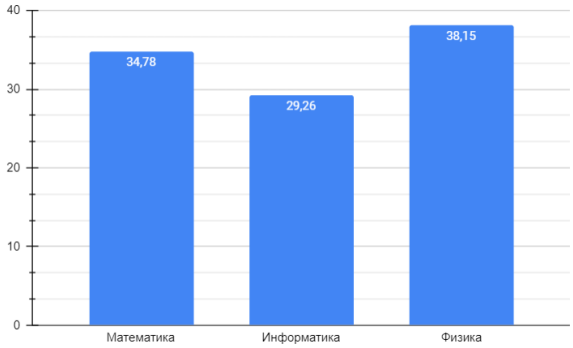
	Excel	**Google Sheets**	**LibreOffice**
Возведение в степень	СТЕПЕНЬ	POW	POWER
Остаток от деления	ОСТАТ	MOD	MOD

Ответ: 31 26

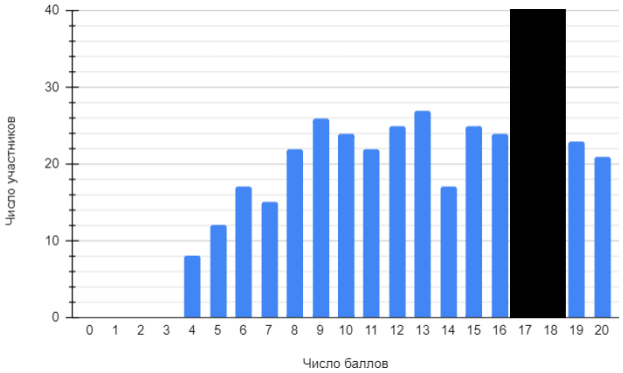
2. Электронные таблицы. Графики и диаграммы (1 балл) [Данные удалены]

Были проведены олимпиады школьников по трём предметам – информатике, математике и физике. В каждой олимпиаде можно было получить от 0 до 20 баллов, чтобы стать призёром нужно было набрать хотя бы 16. По итогам олимпиад организаторы составили две диаграммы: процент призёров среди участников олимпиады и распределение участников олимпиады по математике по числу баллов. Однако часть второй диаграммы была удалена.

Процент призёров среди участников



Распределение участников олимпиады по математике по числу баллов



Определите число участников олимпиады по математике. В ответ запишите одно целое число.
Примечание. Цифры на графике округлены до второго знака после запятой.
Ответ: 368

3. Сортировка и фильтрация данных (3 балла) [Шаги к успеху]

Примечание:

Сортировка пузырьком: выполняется некоторое количество проходов по массиву — начиная от начала массива, перебираются пары соседних элементов массива. Если 1-й элемент пары больше (в случае сортировки по возрастанию; в случае сортировки по убыванию – меньше) 2-го, элементы переставляются (выполняется обмен). Пары элементов массива перебираются (проходы по массиву повторяются) либо n-1 раз (где n – число элементов массива), либо до тех пор, пока на очередном проходе не обнаружится, что более не требуется выполнять перестановки (обмены) (массив отсортирован). При каждом проходе алгоритма по внутреннему циклу очередной наибольший (в случае сортировки по возрастанию; в случае сортировки по убыванию – наименьший) элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом» (в случае сортировки по возрастанию; в случае сортировки по убыванию – наименьшим), а наименьший (в случае сортировки по возрастанию; в случае сортировки по убыванию – наибольший) элемент перемещается на одну позицию к началу массива.

Сортировка слиянием работает по следующему принципу:

1. *Если в рассматриваемом массиве один элемент, то он уже отсортирован — алгоритм завершает работу.*
2. *Иначе массив разбивается на две части, которые сортируются рекурсивно (т.е. на каждой из частей выполняется описанный алгоритм сортировки слиянием).*
3. *После сортировки двух частей массива к ним применяется процедура слияния, которая по двум отсортированным частям получает исходный отсортированный массив. В рамках процедуры слияния сравниваются элементы сливаемых массивов (начиная с начала) и меньший (в случае сортировки по возрастанию) из них записываем в финальный. И затем, в массиве у которого оказался меньший (в случае сортировки по возрастанию) элемент, переходим к следующему элементу и сравниваем теперь его. В конце, если один из массивов закончился, мы просто дописываем в финальный другой массив. После мы наш финальный массив записываем заместо двух исходных и получаем отсортированный участок. С точки зрения исходного массива данная операция атомарна, то есть не имеет промежуточных этапов.*

Пример работы сортировки слиянием:

1. *[4 3 1 6 5] – исходный массив*
2. *[[4 3 1][6 5]] – разделение массивов*
3. *[[[4 3][1]][6 5]] – самый первый неотсортированный подмассив разделяется*
4. *[[[4][3][1]][6 5]] – самый первый неотсортированный подмассив разделяется*
5. *[[3 4][1][6 5]] – производится слияние первых двух одноэлементных массивов. Порядок элементов изменился – шаг 1*
6. *[[1 3 4][6 5]] – производится слияние первых двух отсортированных массивов (результата слияния и одноэлементного). Порядок изменился – шаг 2.*
7. *[[1 3 4][6][5]] – неотсортированный подмассив разделяется*
8. *[[1 3 4][5 6]] – производится слияние одноэлементных подмассивов. Порядок изменился – шаг 3*
9. *[1 3 4 5 6] – производится слияние подмассивов. Порядок элементов не изменился – таким образом, было 3 шага.*

Сегодня на спецкурсе по олимпиадному программированию Илья узнал про два алгоритма сортировки: сортировку пузырьком и сортировку слиянием. Илья стал изучать, как эти сортировки работают на частично отсортированных массивах. Особенно его интересует, какая из сортировок занимает меньше шагов. Шагом Илья называет изменение порядка элементов массива в ходе работы алгоритма. Например, в сортировке пузырьком на массиве [4 2 1 3] по мнению Ильи 4 шага:

1. [4 2 1 3] -> [2 4 1 3]
2. [2 4 1 3] -> [2 1 4 3]

3. [2 1 4 3] -> [2 1 3 4]
4. [2 1 3 4] -> [1 2 3 4]

А в сортировке слиянием на массиве [4 2 3 1] 3 шага:

1. [4 2 3 1] -> [2 4 3 1]
2. [2 4 3 1] -> [2 4 1 3]
3. [2 4 1 3] -> [1 2 3 4]

Помогите Илье определить, какая сортировка сделает меньше шагов на массиве [1 2 3 … 127 128 256 255 … 130 129], состоящем из двух частей – в первой половине натуральные числа от 1 до 128 отсортированы по возрастанию, во второй – числа от 129 до 256 отсортированы по убыванию (таким образом, всего в массиве 256 элементов). В ответе укажите два числа – количество шагов при сортировке массива пузырьком и число шагов при сортировке слиянием.

Ответ: 8128 127

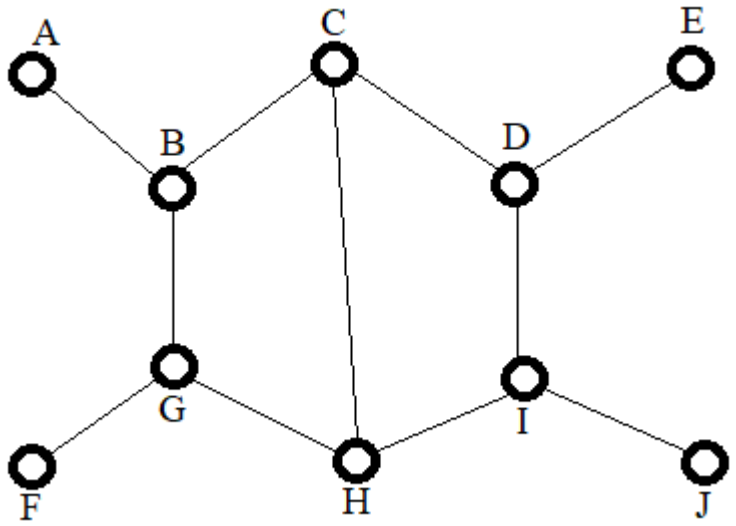
4. Поиск и фильтрация данных (2 балла) [Реакция и хладнокровие]

В некоторой игре персонаж имеет 5 характеристик - Сила, Интеллект, Реакция, Хладнокровие и Техника. Игрок выбирает для персонажа одну либо две характеристики, которые станут основными, одну либо две дополнительных характеристики. Остальные характеристики (от одной до трёх) персонаж не использует. В рамках исследования была проанализированы 1000 различных персонажей. Известно, что персонажей с двумя основными и двумя дополнительными характеристиками оказалось 130, а персонажей с двумя основными и одной дополнительной в 2 раза больше, чем персонажей с одной основной и двумя дополнительными характеристиками. Персонажи с основной характеристикой Реакция и дополнительной Хладнокровие составили 10% от всех персонажей, имеющих по одной основной и дополнительной характеристике. Определите максимальное количество персонажей с основной характеристикой Реакция и дополнительной Хладнокровие при котором такой результат исследования был бы возможен..

Ответ: 84

5. Телекоммуникационные технологии (2 балла) [Файерволы]

В некоторой локальной сети устройства соединены каналами передачи сообщений. Устройства обозначены заглавными буквами английского алфавита.



Соединения между устройствами защищены файерволами, пропускающими сообщение в любом направлении только в том случае, если оно соответствует условию файервола. Ниже приведена таблица условий. Символом m обозначено передаваемое сообщение, & - побитовая конъюнкция, | - побитовая дизъюнкция.

Соединение	**Условие файервола**
AB	m&1000=0
BC	m&11=10
CD	m&111=111
DE	m&1011=1000
BG	m&1=1
CH	m|0=m
DI	m&1000=0
FG	m&101=101
GH	m|10101=10101
HI	m&100=100
IJ	m&1010=0

Сообщения, передаваемые между устройствами, являются натуральными числами, записанными в двоичной системе счисления, возможно, с использованием ведущих нолей. Сообщение передаётся по кратчайшему (проходящему через минимально возможное число соединений) возможному пути. Определите наименьшее возможное двоичное число m, которое можно отправить в сообщении, которое сможет попасть из узла A в узел J и последовательность узлов, через которые сообщение пройдёт (включая начальный и конечный узел). В ответе запишите через пробел сначала двоичное число без ведущих нолей – искомое сообщение, а затем последовательность заглавных английских букв без пробелов – узлы на пути сообщения в порядке, в котором они были посещены. Если сообщение не может быть доставлено укажите NULL.

Пример записи ответа: 10101 ABCDE

Ответ: 101 ABGHIJ

6. Операционные системы (3 балла)

[Чья маска?]

Сэм изучает регулярные выражения. Чтобы применить полученные знания, он написал параметризованный скрипт для поиска файла по маске. Маска задаётся в виде регулярного выражения. В качестве значений параметров могут быть использованы числа или заглавные буквы английского алфавита.

Скрипт имеет следующие параметры:

1. Искать в только текущей папке или в ней и её подпапках (передаётся значение C для поиска в текущей папке и S для поиска в ней и подпапках)
2. Если поиск осуществляется в подпапках, то какая максимальная глубина поиска (натуральное число - если глубина ограничена, буква A – если ограничения нет; число 1 соответствует поиску только в текущей папке)
3. После скольки совпадений остановить поиск (натуральное число - если необходимо ограничить число найденных совпадений, буква A – если ограничения нет). В случае, если не удастся найти заданное число совпадений, будут показаны те, что удалось найти.
4. Регулярное выражение, по которому осуществляется поиск. Формат регулярного выражения описан ниже.

Для задания регулярных выражений приняты следующие обозначения:

c Любой неспециальный символ c соответствует самому себе. Специальными символами будем считать только символы [,], {, }, *, +, -, ? – эти символы не могут по условию данной задачи встретится в тексте.

[...] Любой символ из ...; допустимы диапазоны типа a-z (последовательно идущие символы в алфавите); возможно объединение диапазонов, например [a-z0-9] и сочетание диапазонов и отдельных символов [a-z0-9~#].

r* Ноль или более вхождений символа r, может применяться и для диапазонов, например [a-z#]* означает ноль или более вхождений любых символов из диапазона от a до z или символа # в любом порядке.

r+ Одно или более вхождений символа r, может применяться и для диапазонов, например [a-z>]+ означает одно или более вхождений любых символов из диапазона от a до z или символа > в любом порядке.

r? Ноль или одно вхождение символа r, может применяться и для диапазонов, например [a-z@]? означает ноль или одно вхождение любого символа из диапазона от a до z или символа @.

r1r2 За символом или диапазоном r1 следует символ или диапазон r2.

{ } Число вхождений предыдущего выражения. Например, выражение [0-9]{5} соответствует подстроке из пяти десятичных цифр.

^ Символ начала строки. Регулярное выражение должно начинаться с этого символа.

$ Символ конца строки. Регулярное выражение должно заканчиваться этим символом.

Пример: регулярное выражение ^a+[a-z]{5}.[0-9]*$ позволяет найти все последовательности символов, которые начинаются с одного или нескольких символов a, после которых идут ровно 5 маленьких латинских букв, затем точка и затем может следовать любое количество (в том числе ноль) арабских цифр.

Фрагмент файловой системы, на котором он проверял скрипт выглядит следующим образом:

folder1	**folder2**	**folder3**	**folder4**	**folder5**
folder2	folder3	folder5	vhs.exe	balloon.mp4
baobab.png	folder4	bavaria.mov	qsort.cpp	bunny_baks.jpg
abba_song.mp3	blablacar.zip	baldurs.mp4	mergesort.java	abcabcabc.doc
albania.zip	lalaland.mov	black.png	algorythm.java	
the_abs.doc	absolute.cpp	band.png	sort_algo.jar	
car_ab.jpg			alter_ego_a.mp4	
vabadab.doc			tart.zip	
the_asbolute.mov			alter_ego_b.mp4	

В первой строке каждого столбца указано название папки, далее – её содержимое. Все элементы с названием folder*, где * является числом, являются папками. В момент запуска скрипта обработка начинается с содержимого folder1 в алфавитном порядке. Сначала скрипт проверяет файлы, затем содержимое подпапок. В случае нескольких подпапок они также проверяются в алфавитном порядке.

Скрипт был запущен с параметрами S, 2, 5 и некоторым регулярным выражением и вывел следующий результат:

albania.zip
abba_song.mp3
baobab.png
car_ab.jpg
the_abs.doc

Определите, какие из представленных регулярных выражений могли быть использованы:

1. ^[a-z_]*a[a-z_]*a[a-z0-9.]*$
2. ^a[a-z]*$
3. ^[a-z_]*a[a-z0-9._]*$
4. ^[a-z0-9._]{6}[a-z0-9._]*$
5. ^[a-z_]*a[a-z.]*$
6. ^[a-z0-9._]+$
7. ^[a-z_]*a[a-z0-9]*$

В ответ запишите через пробел в порядке возрастания номера всех регулярных выражений, которые могли обеспечить подобный результат.

Примечание: Глубиной поиска в файловой системе называют число папок в пути от корневой до текущей, включая корень. Например, глубина поиска в корневой папке – 1, т.к. путь до неё содержит всего одну папку – её саму. Если текущая папка лежит в корневой папке – её глубина равна 2, и так далее. Количеством совпадений (количеством результатов поиска) называют число найденных файлов, которые удовлетворяют условиям поиска.

Ответ: 3 4 6

7. Технологии программирования (2 балла) [Рассадка по аудитории]

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

В начале учебного года на урок программирования пришли k школьников. Места для посадки в аудитории разбиты на n рядов, пронумерованных от 1 до n. В каждом ряду m мест, пронумерованных от 1 до m. Поскольку это начало учебного года, необходимо в первую очередь определиться с рассадкой.

По различным причинам (по состоянию здоровья или просто потому что захотелось), у некоторых школьников есть ограничения по тому, где они могут сидеть. Ограничения бывают двух видов:

школьник номер s_i может сидеть только на рядах с $r_{i,1}$ по $r_{i,2}$;

школьник номер s_i может сидеть только на местах с номерами с $c_{i,1}$ по $c_{i,2}$.

Всего таких ограничений ровно p. Для одного и того же школьника может быть не более одного ограничения каждого из двух видов. Назовем школьника *недовольным*, если ему не было выдано место в аудитории, либо если хотя бы одно из его ограничений не выполнено.

Изначально ни одному школьнику не было назначено место. Затем преподаватель q раз отдавал поручения вида «школьнику s_i сесть на место c_i в ряду r_i». Если при этом

или это место уже было кем-то занято,

или школьник s_i **уже сидел на каком-то другом месте**, а при такой пересадке количество невыполненных ограничений **строго увеличивалось**,

поручение игнорировалось, и школьник s_i оставался на месте. Обратите внимание, что если у школьника не было назначенного ему места, а назначенное место свободно, поручение будет выполнено вне зависимости от ограничений школьника.

Обработайте все поручения преподавателя и выведите в конце количество недовольных школьников.

Формат входных данных

Решите задачу для t различных классов. В первой строке ввода дано целое число t — количество классов, для которых требуется решить задачу ($1 \le t \le 100$). Затем следуют описания классов и последовательностей перемещений школьников.

В первой строке описания класса через пробел даны четыре целых числа n, m, k и p — размеры аудитории, количество школьников и количество ограничений ($1 \le n, m \le 100$, $1 \le k \le 1000$; $0 \le p \le 2k$).

Каждая из следующих p строк содержит ограничение в формате «R s_i $r_{i,1}$ $r_{i,2}$» или «C s_i $c_{i,1}$ $c_{i,2}$» — либо ограничение на номер ряда, либо ограничение на номер места для школьника s_i ($1 \le s_i \le k$; $1 \le r_{i,1} \le r_{i,2} \le n$; $1 \le c_{i,1} \le c_{i,2} \le m$). Гарантируется, что у каждого школьника есть не более одного ограничения каждого вида.

В следующей строке дано целое число q — количество поручений преподавателя ($1 \le q \le 5000$). Каждая из следующих q строк описывает поручение преподавателя в формате «MOVE s_i r_i c_i» — школьнику s_i пересесть в ряд r_i на место c_i ($1 \le s_i \le k$; $1 \le r_i \le n$; $1 \le c_i \le m$).

Формат выходных данных

Для каждого из t классов выведите в отдельной строке единственное целое число — количество недовольных школьников в классе после обработки всех поручений преподавателя.

Пример

Стандартный ввод	Стандартный вывод
3 2 2 4 0 3 MOVE 1 1 1 MOVE 2 1 1 MOVE 3 1 1 2 2 4 2 R 1 2 2 C 3 1 1 4 MOVE 1 1 1 MOVE 2 1 2 MOVE 3 2 1 MOVE 4 2 2 3 3 100 2 R 1 1 2 R 2 2 3 5 MOVE 1 2 3 MOVE 2 1 2 MOVE 100 1 1 MOVE 99 1 1 MOVE 98 2 2	3 1 97

8. Технологии программирования (4 балла)

[Космические шашки]

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

Как вам известно, находясь на орбите, космонавтам абсолютно нечего делать. Наши герои, представители небезызвестной династии покорителей космоса Шелби: Валентиныч и Валерьич, решили взять с собой набор для игры в космические шашки, дабы не заскучать во время экспедиции Voyager-11.

Космические шашки представляют собой прямоугольное поле из n строк и m столбцов, каждая клетка которого может быть пустой или содержать либо черную, либо белую шашку. Строки пронумерованы от 1 до n сверху вниз, столбцы — от 1 до m слева направо.

Ходы делаются по очереди, ход заключается в передвижении своей шашки на незанятое поле одним из двух способов:

- Если есть свободная вперед по диагонали клетка, шашка может быть передвинута на одну клетку по диагонали. Белые могут совершить такой ход только по направлению **уменьшения** номера строки, а черные — по направлению **увеличения** номера строки.
- Если на соседней с шашкой диагональной клетке находится шашка соперника, и за ней имеется свободное поле, шашка может быть передвинута на следующую по диагонали клетку за шашкой соперника (при этом шашка соперника снимается с поля). Если после этого хода имеется продолжение для взятия других шашек соперника, ход может быть продолжен. Такой ход может производиться в любом из четырех направлений по диагонали.

Иными словами, можно либо сделать ход на одну клетку по диагонали, после чего остановиться, либо сразу начать брать шашки противника в произвольном количестве. Но нельзя сначала сделать ход первым способом, а затем продолжить его вторым.

При совершении очередного хода шашка игрока может стать *дамкой*. Белая шашка становится дамкой, если доходит до 1-й строки игрового поля, а черная — если доходит до n-й строки игрового поля.

В очередной день, играя долгую партию, Валентиныч задался вопросом: а может ли сейчас какая-то шашка за один ход стать дамкой? Зная текущее расположение шашек на поле и чей сейчас ход, помогите Валентинычу найти ответ на его вопрос.

Формат входных данных

Требуется найти ответ на задачу для t различных партий. В первой строке ввода дано целое число t — количество партий, для которых надо решить задачу ($1 \le t \le 100$). Далее следуют t описаний текущего состояния партии.

В первой строке каждого описания партии через пробел даны два целых числа n и m — размеры поля, а также символ ‘w’, если сейчас ход белых, и ‘b’, если ход черных ($1 \le n, m \le 1000$).

Следующие n строк описывают поле; i-я строка ввода описывает i-ю строку поля. Каждая строка содержит ровно m символов, каждый из которых может быть либо ‘.’, если соответствующая клетка свободна, либо ‘W’, если на клетке расположена белая шашка, либо ‘B’, если на клетке расположена черная шашка.

Гарантируется, что сумма размеров полей ($n \cdot m$) по всем t партиям не превосходит 10^6.

Формат выходных данных

Для каждой партии выведите в отдельной строке через пробел два целых числа — номер строки и номер столбца, в которых расположена шашка, которая за ближайший ход может выйти в дамки.

Если такой шашки нет, выведите «$-1\ -1$» (без кавычек). **Если таких шашек несколько, выберите шашку, расположенную на строке с минимальным номером, а из всех таких — в столбце с минимальным номером.**

Пример

Стандартный ввод	Стандартный вывод
8 5 5 w ....B.. ....B.. .W.W. 8 8 w ..... BBBBBBBB .B.B.B.B BBB.B.B. .W.W.W.W .B..... ..W..... ..... 10 9 b .....B.B ....B.... .BB.B.BW. ..W..... WBW.W.BB. ..BBWBBWB B.....BB. .B..WBB.. .....B... ..... 8 9 w ..... BB.BBB..B .WW.....B .....W.. .... .B..... ..... 2 7 w B.BB... ...WW.. 7 2 b .. .. .. .W .W W. .. 2 8 b .B.BB... ..... 5 5 w	5 2 7 3 9 6 3 3 2 4 -1 -1 1 2 -1 -1

Задания для 5–8 класса

Заключительный этап (приведен один из вариантов заданий)

1. Кодирование информации, информационный объем (1 балл)

[Старый плеер]

Даша нашла на старом плеере три музыкальные композиции и заметила интересную особенность: все композиции имеют одинаковую частоту дискретизации, однако глубина кодирования первой композиции в 2 раза меньше глубины кодирования второй композиции, а глубина кодирования третьей – в 3 раза больше глубины кодирования второй. Тогда Даша придумала задачку и предложила ее решить своей подруге Алисе: определить длительность третьей композиции. Все композиции двухканальные, а также известно, что первая и третья композиции имеют одинаковый информационный объём, а длительность первой композиции равна 120 секундам. Помогите Алисе решить задачку Даши. В ответ запишите одно целое число – длительность третьей композиции в секундах.

Ответ: 20

Решение:

Применим формулу $I = D \cdot t \cdot i \cdot k$. Обозначим за x глубину кодирования первой композиции. Тогда 2x – глубина кодирования второй композиции, а $3 \cdot 2x$ – глубина кодирования третьей композиции.

$$I_1 = D \cdot t \cdot i \cdot k = D \cdot 120 \cdot x \cdot 2$$
$$I_2 = D \cdot t \cdot i \cdot k = D \cdot t_2 \cdot 2x \cdot 2$$
$$I_3 = D \cdot t \cdot i \cdot k = D \cdot t_3 \cdot 3 \cdot 2x \cdot 2$$

Применим то, что 1я и 3я композиции имеют одинаковый информационный объем.

$$I_1 = I_3$$
$$D \cdot 120 \cdot x \cdot 2 = D \cdot t_3 \cdot 3 \cdot 2x \cdot 2$$

Решим уравнение.

$$120 = t_3 \cdot 6$$
$$20 = t_3$$

2. Комбинаторика (2 балла)

[Шахматы 2: Спасти рядового Пешку]

Дана шахматная доска размера 16x16 клеток и фигура, которая может ходить только по диагонали, но со следующими ограничениями:

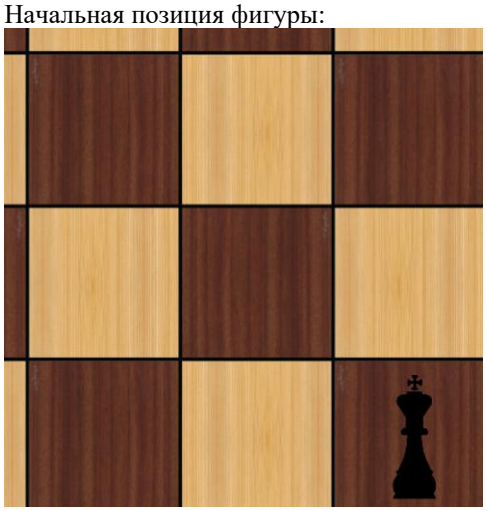
1) Фигура не может ходить по диагонали вправо-вниз.
2) На ходах с нечётным номером фигура может пойти **только** влево-вверх.
3) На ходах с чётным номером фигура может выбрать **одно из двух** направлений:

a. Пойти вправо-вверх.
b. Пойти влево-вниз.

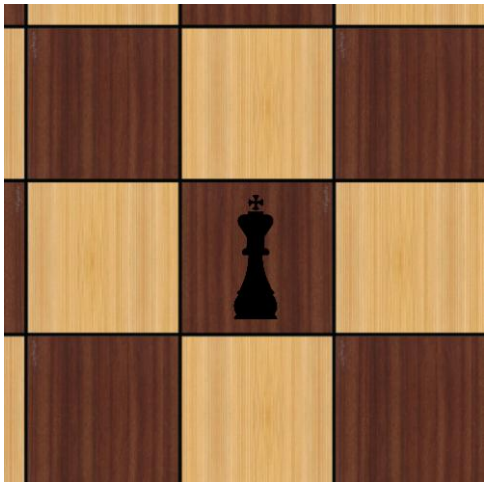
4) Нумерация ходов начинается с единицы.

Пример:

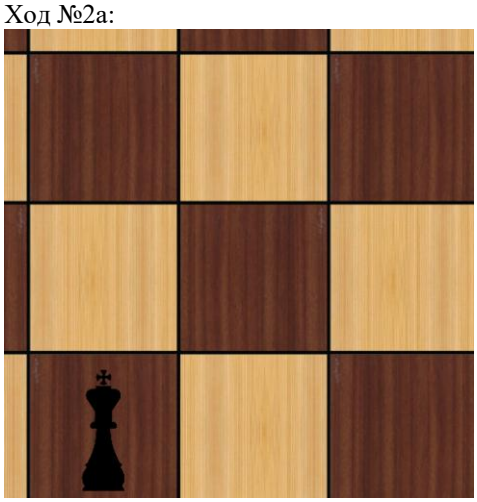
Начальная позиция фигуры:



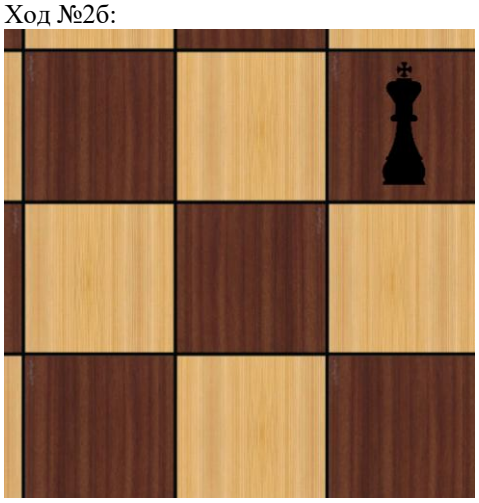
Ход №1:



Ход №2а:

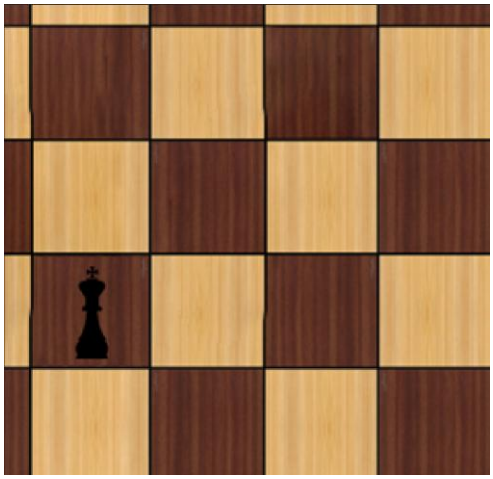


Ход №2б:



Ходы №2а и №2б являются взаимоисключающими.

Ход №3а (из позиции хода №2а):



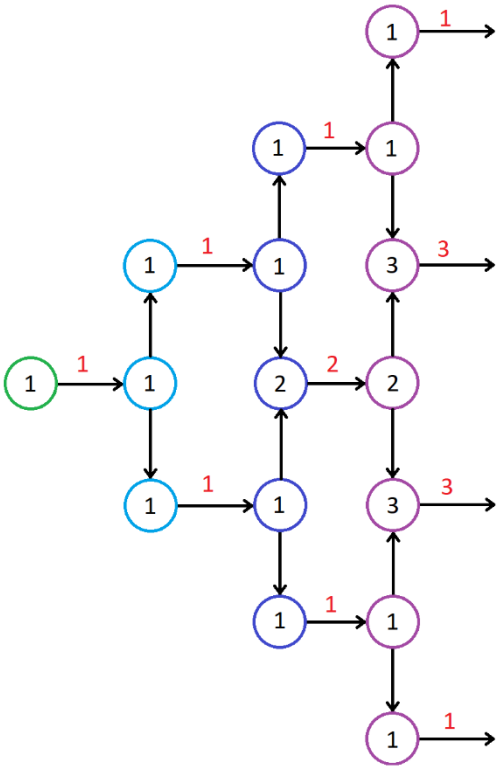
Ход №3б (из позиции хода №2б):

Сколько существует различных траекторий, отличающихся хотя бы одним ходом, для того, чтобы добраться из правого-нижнего угла доски в левый-верхний угол? Число возможных траекторий напишите в ответ.

Ответ: 3432

Решение:

Сначала для удобства представим несколько первых переходов между клетками в виде графа



Каждый цвет представляет первые несколько отдельных диагоналей (параллельные диагонали из левого-нижнего угла в правый-верхний)

Для первых нескольких диагоналей посчитаем количество возможных переходов на следующую диагональ. Заметим, что количество переходов из клеток образуют треугольник Паскаля. Треугольник так и будет расти до момента, когда мы дойдем до самой длинной диагонали в середине доски. Легко понять, что при размере доски 16x16 клеток, длина самой длинной диагонали из левого нижнего края в правый верхний, на которую мы можем встать, будет равна 15 клеткам. Соответственно для того, чтобы дойти из угла до диагонали потребуется 7 переходов между диагоналями из начальной позиции. Значит нам нужна 8 строка в треугольнике Паскаля. Заметим, что после того, как мы дошли до самой длинной диагонали белого цвета, крайние клетки диагонали начинают нам не подходить, так как из них нет путей к следующим диагоналями, и с каждым переходом нужно «отбрасывать» по два крайних числа в строке. И продолжать так делать, пока не останется одно число:

8 строка в треугольник Паскаля:

1 7 21 35 35 21 7 1

Получим следующую строку и сразу отбросим крайние члены

8 28 56 70 56 28 8

Получим следующую строки и вновь отбросим члены

36 84 126 126 84 36

Продолжим применять этот алгоритм:

120 210 252 210 120

330 462 462 330

792 924 792

1716 1716

В итоге складываем последние два числа и получаем **3432**

3. Системы счисления. Теория игр (2 балла) [Игра в числа]

Прохор и Василиса придумали игру с числами.

В начале игры на доске записывается некоторое стартовое число в десятичной системе счисления. Далее игроки ходят по очереди. Каждый игрок в свой ход стирает число на доске и записывает вместо него запись этого числа в восьмеричной или в девятеричной системе счисления (на свой выбор). Запись на доске следующим игроком читается как новое десятичное число. Выигрывает игрок, записавший в свой ход на доске «300» или большее число. Первым ходит Прохор. Какое минимальное стартовое число должно было быть на доске, чтобы гарантированно выиграл Прохор своим вторым ходом? В ответ запишите искомое число в десятеричной системе счисления.

Пример игры, если бы стартовым числом было 100, а для победы было необходимо записать число, равное или большее 200:

Прохор своим первым ходом переводит 100 в восьмеричную систему счисления и заменяет число на доске на 144. Василиса своим первым ходом переводит 144 в девятеричную систему счисления и заменяет число на доске на 170. Прохор своим вторым ходом переводит 170 в девятеричную систему счисления, заменяет число на доске на 208 и побеждает.

Ответ: 114

Решение:

Заметим, что перевод в 8 систему счисления всегда даст большее число, чем перевод в 9 систему счисления.
Переведем число 300 из 8 системы счисления.

$$300_8 = 192_{10}$$

Таким образом, если после первого хода Василисы на доске останется любое число, равное или большее 192, то Прохор победит, переведя его в восьмеричную систему счисления. Василиса в свой ход может переводить предыдущее число как в восьмеричную, так и в девятеричную системы счисления. Так как в задаче спрашивается минимальное число, необходимое для гарантированной победы Прохора, будем считать, что Василиса будет переводить в свой ход число в 9 систему счисления, но будем следить за тем, чтобы перед ходом Василисы не было выигрышной позиции (192 или больше).
Какое число при переводе в девятеричную систему счисления даст число 192? Никакое, так как цифры 9 в девятеричном числе быть не может. Какое минимальное число, большее 192, может быть переведено из девятеричной системы в десятичную? 200. $200_9 = 162_{10}$. Таким образом, минимальное число, которое можно оставить для Василисы на начало ее хода – 162. Тогда минимальное стартовое число, которое Прохор мог превратить в 162 – это 114 (переводом в восьмеричную систему счисления).
Проверим, что при стартовом числе 114 Прохор действительно выиграет независимо от ходов Василисы:
114 -> (Прохор переводит число в 8 систему счисления) -> 162 -> (Василиса переводит число в 9 систему счисления) -> 200 -> (Прохор переводит число в 8 систему счисления) -> 310, Прохор побеждает.
114 -> (Прохор переводит число в 8 систему счисления) -> 162 -> (Василиса переводит число в 8 систему счисления) -> 242 -> (Прохор переводит число в 8 систему счисления) -> 362, Прохор побеждает.
Проверим также, что при меньшем 114 числе (113) Прохор не сможет выиграть вторым ходом при правильном ходе Василисы.
113 -> (Прохор переводит число в 8 систему счисления) -> 161 -> (Василиса переводит число в 9 систему счисления) -> 188 -> (Прохор переводит число в 8 систему счисления) -> 274, Прохор не побеждает вторым ходом. (Рассматривать варианты ходов Прохора, где он переводит числа в девятеричную систему счисления не имеет смысла, так как это не увеличит итоговое число)

4. Основы логики (3 балла)
[Булева функция]

Петя придумал следующую булеву функцию:

$$\big(p \to (\text{НЕ}(q)\ \text{ИЛИ}\ r)\ \text{ИЛИ НЕ}(r)\big)\ \text{И}\ (p\ \text{И НЕ}(q \to r)\ \text{ИЛИ}\ \big(q \to (\text{НЕ}(p)\ \text{ИЛИ}\ r)\big))$$

Также Петя знает о существовании функций, обладающих следующим свойством:

$$f(\text{НЕ}(x_1), \dots, \text{НЕ}(x_n)\ ) = \text{НЕ}(f(x_1, \dots, x_n))$$

Теперь Петя хочет узнать, сколько у него есть способов сделать из своей функции новую, обладающую этим свойством. Новую функцию Петя может сделать следующим образом: записать новую переменную, затем логическую операцию, затем придуманную им исходно функцию в скобках. Причем новая переменная должна быть из следующего множества: {p, q, r, НЕ(p), НЕ(q), НЕ(r)}, а логическая операция из множества: {И, ИЛИ, →}.

Пример: если изначальная функция была ***p ИЛИ q****, то одним из примеров новой функции будет:* ***<u>p И</u> (p ИЛИ q).***

В ответе нужно записать целое число – количество способов составления новой функции, обладающей указанным свойством.

Примечание: → - ИМПЛИКАЦИЯ, логическая операция, таблица истинности которой выглядит следующим образом:

p	q	p→q
0	0	1
0	1	1
1	0	0
1	1	1

Ответ: 6

Решение:

Исходя из описанного свойства, заметим, что функция должна принимать противоположные значения на противоположных значениях ее аргументов (т.е. функция самодвойственна).

Если мы будем смотреть на таблицу истинности такой функции, то значения, полученные при прочтении ее сверху вниз до середины и при прочтении снизу вверх до середины должны быть противоположны друг другу.

Составим таблицу истинности для записанного Петей выражения:

P	q	r	**f**
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Замечаем, что данное выражение является тождеством. Следовательно, при добавлении в начало логической операции и переменной результат полученного выражения будет во многом зависеть от новой переменной.

Однако, если в качестве добавляемой логической операции мы будем использовать «логическое или» или «импликацию» с любой переменой, то полученное выражение у нас останется тождеством, т.к. (***x*** or 1 = 1) и (***x*** → 1 = 1) – на месте ***x*** может стоять любая из переменных, принадлежащих множеству допустимых переменных из задачи. Тогда остается логическое И.

Если рассматривать «логическое и», то в этом случае результат функции будет совпадать со значением добавленной переменной, если в качестве переменной будет выбрана одна из {p, q, r}, либо будет противоположно значению добавленной переменной, если в качестве новой переменной будет выступать одна из {НЕ(p), НЕ(q), НЕ(r)}. И если посмотреть на составленную таблицу истинности, то можно заметить, что значения переменных в столбцах при прочтении снизу вверх и сверху вниз как раз противоположны друг другу, причем у всех трех переменных. Следовательно новая функция

x И $\Big(\big(p \to (\text{НЕ}(q)\ \text{ИЛИ}\ r)\ \text{ИЛИ НЕ}(r)\big)\ \text{И}\ \Big(p\ \text{И НЕ}(q \to r)\ \text{ИЛИ}\ \big(q \to (\text{НЕ}(p)\ \text{ИЛИ}\ r)\big)\Big)\Big)$ будет обладать нужным свойством, где в качестве x стоит одна из допустимых переменных. Итого вариантов составить нужную функцию будет 6.

5. Кодирование информации, структуры данных (3 балла)
[Шифрование на графе]

Бобу нужно было передать пароль двум друзьям. Известно, что пароль состоит только из букв русского алфавита, причем в пароле нет повторяющихся букв.

Боб не хочет, чтобы его пароль узнал кто-то посторонний, поэтому придумал для каждого из друзей различные шифры – «шифр в глубину» и «шифр в ширину».

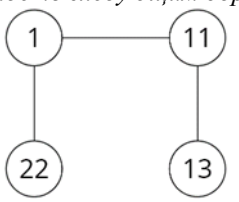
Но для построения шифра нужно сначала построить граф пароля. Строится он следующим образом:

1. Каждая буква в пароле заменяется на число от 1 до 33 в соответствии с ее порядковым номером в русском алфавите.

2. Строится полный граф (т.е. граф, в котором каждая вершина соединена с каждой), в котором в качестве вершин выступают полученные числовые значения букв пароля.

3. Из полученного графа удаляются ребра, которые соединяют между собой те числа, буквенные значения которых НЕ стоят рядом в пароле.

Пример: «ФАЙЛ» - исходный пароль => «22 1 11 13» - численное представление. Тогда граф исходного пароля будет выглядеть следующим образом:

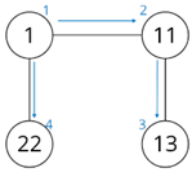


Теперь по этому графу можно построить два шифра. Для построения обоих шифров в качестве начальной вершины выбирается вершина с наименьшим номером. Соседями текущей вершины называются такие вершины, которые соединены с текущей вершиной ребром.

Для построения «шифра в глубину» необходимо:

0. Изначально все вершины графа считаются не посещенными.
1. В качестве первой текущей вершины выбрать вершину с наименьшим числом и записать ее. Считаем эту вершину посещенной.
2. Далее из соседей текущей вершины выбирается вершина с наименьшим номером и записывается ее номер. Переходим в эту вершину (т.е. она становится текущей) и считаем эту вершину посещенной.
3. Повторяем действие 2, пока не встретим вершину, у которой нет не посещённых соседей.
4. Переходим к первой посещенной вершине и повторяем действие 2 для не посещенных соседей.
5. Далее переходим ко второй посещенной вершине и повторяем действие 2 для не посещенных соседей.
6. И так далее идем по посещенным вершинам в порядке их посещения и повторяем действие 2 для не посещенных соседей, пока в графе все вершины не станут посещенными.

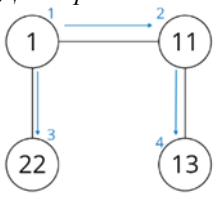
Для пароля «ФАЙЛ» построение и сам «шифр в глубину» будет выглядеть следующим образом: *1 11 13 22*



Для построения «шифра в ширину» необходимо:

0. Изначально все вершины графа считаются не посещенными.
1. В качестве первой текущей вершины выбрать вершину с наименьшим числом и записать ее. Считаем эту вершину посещенной.
2. Далее записываем всех ее не посещенных соседей в порядке возрастания чисел, записанных в этих вершинах. В этом же порядке считаем эти вершины посещенными.
3. Далее идем по посещенным вершинам (в порядке их посещения) и повторяем действие 2.
4. Идем так, пока в графе все вершины не станут посещенными

Для пароля «ФАЙЛ» построение и сам «шифр в ширину» будет выглядеть следующим образом: *1 11 22 13*



Ева смогла получить оба шифра Боба и узнать какой шифр соответствует какому алгоритму:

Шифр в глубину: 10 14 16 25 20 33

Шифр в ширину: 10 14 20 16 33 25

Также Ева узнала, что пароль начинается с буквы Я.

Помогите Еве разгадать пароль Боба.

В ответ запишите пароль заглавными буквами без пробелов.

Примечание: русский алфавит:

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я			

19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

Ответ: ЯТИМОЧ

Решение:

Заметим, что так как буквы в пароле не повторяются, и в окончательном графе пароля остаются только ребра, соединяющие соседние буквы, то тогда у каждой вершины может быть либо один, либо два соседа (под соседями подразумеваются соседние по ребрам вершины)

Рассмотрим два случая, когда у вершины 10 будет 1 сосед и когда 2 соседа:

1) Если у вершины 10 один сосед.

Тогда, если посмотреть на шифр в ширину, то по его алгоритму соседом вершины 10 будет только вершина 14. И так как у вершины 10 больше нет соседей, то по алгоритму далее должна рассматриваться вершина 14 и получается, что у нее тоже только один сосед – вершина 20 (два соседа у нее не могут быть, так как далее идет число 16, а если бы вершина 16 была соседом вершины 14, то в шифре 16 стояло бы до 20).

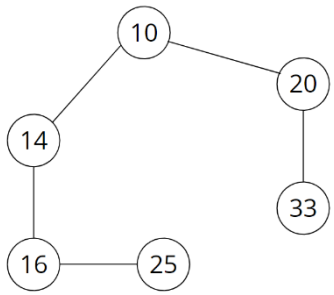
Однако если посмотреть на шифр в глубину, то сначала получается, что у вершины 14 есть сосед 16, что не сходится с первым шифром. Противоречие.

2) Если у вершины 10 два соседа.

Смотря на шифр в ширину, понимаем, что соседи вершины 10 – вершина 14 и вершина 20.

Далее по шифру в глубину заметим, что у вершины 14 соседом является вершина 16. И так как у вершины 14 уже два соседа, то из шифра в ширину получаем, что вершина 33 является соседом вершины 20, и тем самым и у вершины 20 получаются два соседа, тогда оставшаяся вершина 25 является соседом вершины 16, что можно заметить из шифра в глубину.

Тогда получается, что граф численного представления пароля будет выглядеть следующим образом:



В начале мы заметили, что в оставшемся графе ребрами соединены вершины, чьи буквенные эквиваленты стоят рядом в пароле, то тогда можно восстановить пароль последовательно пройдясь по получившейся цепочке. Остается только узнать каким буквам соответствуют данные порядковые номера и с какого конца нужно будет начинать считывание, но так как в задаче сказано, что пароль начинается с буквы Я, порядковый номер которой 33, то тогда с этой вершины и следует начинать дешифрование.

Тогда численный пароль будет: 33 20 10 14 16 25, а буквенный пароль ЯТИМОЧ

6. Теория игр (2 балла)

[Весенняя уборка]

Однажды, Муми-тролль вместе со Сниффом решили убраться в доме после долгой зимней спячки. К своему удивлению, они нашли так много грецких орехов, что сложили из них целую кучку. С этой кучкой они решили сыграть в следующую игру:

Игра состоит из раундов. В каждом раунде сначала ходит первый игрок, а потом второй. Каждый из игроков в свой ход может взять из кучи либо 7 орехов, либо t орехов (или все оставшиеся, если орехов в куче меньше 7 или меньше t). Игрок, после хода которого, в кучке не осталось орехов считается победителем.

В первом раунде t = 2. Но с каждым следующим раундом t = t · 2. То есть во втором раунде t = 4, в третьем t = 8 и так далее.

Определите, какое минимальное число орехов изначально могло быть в кучке, чтобы гарантированно победил 1-ый игрок и при том на четвёртом раунде.

Ответ: 36

Решение

Если игра была закончена первым игроком и при том на четвёртом раунде, то к началу четвёртого раунда в куче должно было быть от 1 и до 16 орехов.

Тогда в третьем раунде (t = 8): Перед ходом второго игрока в кучке должно быть (1 + 8 = 9; 16 + 7 = 23) от 9 и до 23 орехов, а перед ходом первого игрока в третьем раунде в кучке должно быть (9 + 7 = 16; 23 + 8 = 31) от 16 и до 31 орехов.

Во втором раунде (t = 4): Перед ходом второго игрока в куче должно быть (16 + 7 = 23; 31 + 4 = 35) от 23 и до 35 орехов. Перед ходом первого игрока в куче должно быть (23 + 4 = 27; 35 + 7 = 42) от 27 и до 42 орехов.

В первом раунде (t = 2): Перед ходом второго игрока в куче должно быть (27 + 7 = 34; 42 + 2 = 44) от 34 и до 44 орехов. Перед ходом первого игрока в куче должно быть (34 + 2 = 36; 44 + 7 = 51) от 36 и до 51 орехов.

По сути, в этих вычислениях итеративно изменяется числовой интервал возможного числа орехов. Для второго игрока нижняя граница интервала увеличивается на максимальное число из чисел 7 и t, чтобы второй игрок своим ходом не мог попасть ниже нижней границы интервала. Аналогично верхняя граница увеличивается на минимальное число из чисел 7 и t.

А для первого игрока всё наоборот, нижняя граница увеличивается на минимальное из 7 и t, а верхняя на максимальное из 7 и t. В этой задаче верхнюю границу мы могли и не учитывать, чтобы получить верный ответ.

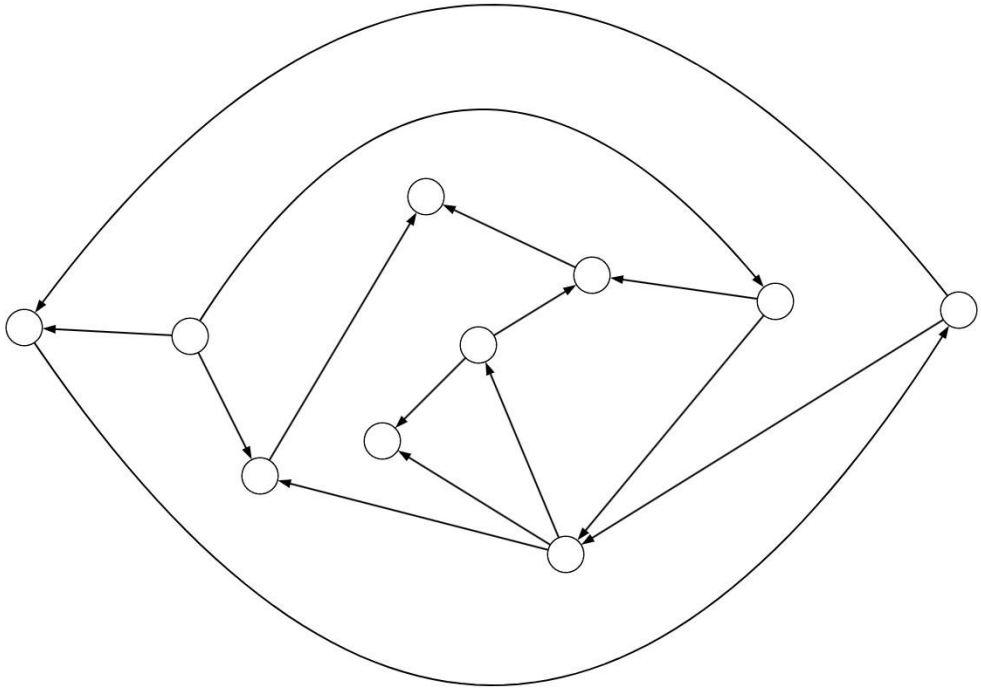
Минимальное изначальное число орехов, при котором победил первый игрок на четвёртом раунде – это 36

7. Моделирование на графе (1 балл) [Музейная инновация]

Архитекторы решили оснастить музейные залы самодвижущимися дорогами - траволаторами. По такой дороге возможно двигаться лишь в одном направлении. К сожалению, из-за просчётов в плане, некоторые из дорог развернули не в ту сторону. Вы можете поменять направление каких-то дорог так, чтобы независимо от того, из какой комнаты посетитель попадает на траволатор, он мог добраться по траволатору до любой комнаты.

План музея представляет собой ориентированный граф, где комнаты обозначаются вершинами, а дороги траволатора – рёбрами. При перемещении по музею, по траволатору можно двигаться только в направлении указанному стрелкой.

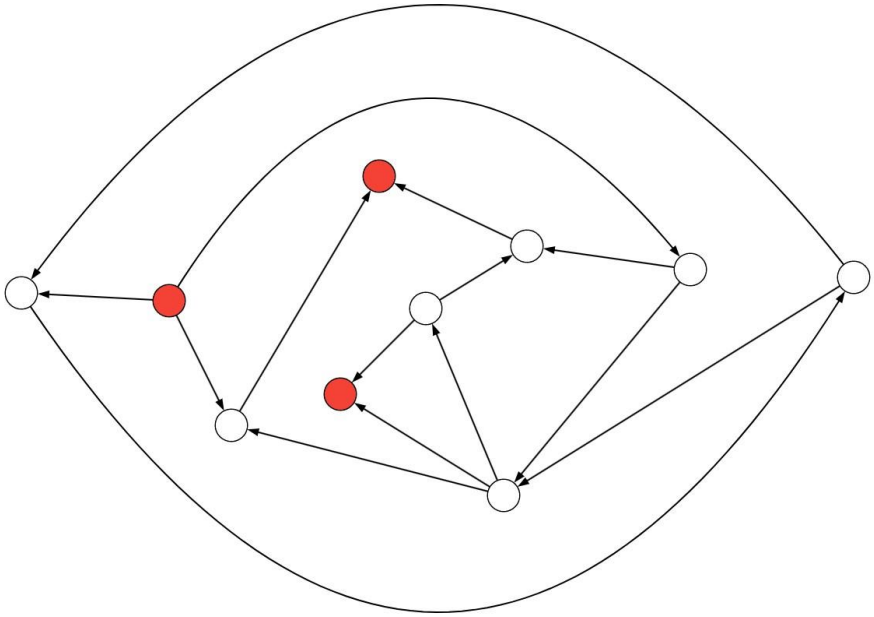
Требуется исправить план музея так, чтобы число перенаправленных дорог было минимально. В ответе укажите число перенаправленных дорог.



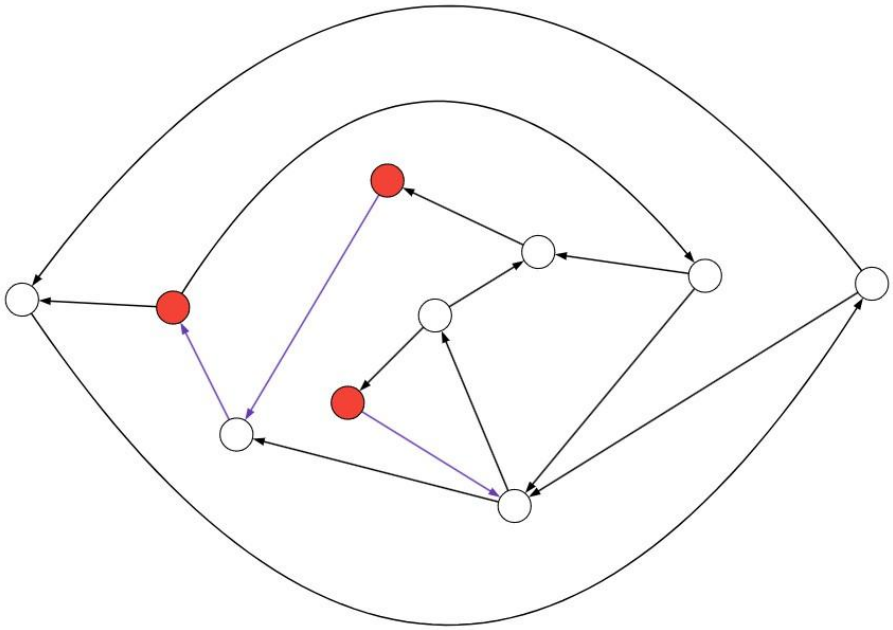
Ответ: 3

Решение:

Для начала, отметим такие вершины, у которых все рёбра имеют одинаковую направленность, т.е. либо все рёбра исходят из вершины, либо все рёбра входят в вершину. Назовём такие вершины неправильными.



Для того, чтобы из любой вершины был путь в любую другую, нужно как минимум для каждой из выбранных вершин изменить направление как минимум одного соседнего с ней ребра так, чтобы такой выбор ребер не создал новых неправильных вершин – так как в неправильные вершины либо невозможно попасть, либо невозможно из них выйти. Возможный выбор ребер:



Заметим, что теперь желаемая цель достигнута, из любой вершины можно попасть в любую другую, следовательно дополнительных изменений не требуется.

8. Моделирование (1 балл)
[Алгоритм на таблице]

Расстоянием между числами является модуль их разности, т.е. разность между большим и меньшим числом.

Ближайшим числом к текущему числу называется такое число, что расстояние между текущим и этим числом минимально.

Дана матрица размера 10x10, заполненная числами от 1 до 2^8:

40	151	91	201	127	188	167	26	47	60
86	201	173	61	33	67	3	178	57	57
116	139	187	174	252	209	184	27	84	152
85	250	103	238	97	103	84	188	50	251
158	243	33	182	90	115	46	164	111	100
45	88	48	244	130	25	220	135	195	190
12	114	69	76	176	52	206	220	117	49
157	91	74	104	154	243	156	113	37	66
166	74	210	188	43	115	41	29	171	129

224	221	165	60	95	200	39	39	21	181

К ней применятся следующий алгоритм:

1. Из отрезка [1, 2^8] выбираются 2^{8-i} чисел, таких что в отсортированном виде все соседние пары чисел имеют одинаковое расстояние. Числа 1 и 2^8 входят в этот список. Выбранные числа не обязательно должны быть целыми.
2. Каждое число в матрице заменяется на ближайшее число из выбранных на первом шаге чисел. Если у числа в матрице два ближайших числа из выбранных на первом шаге, то оно заменяется на меньшее из ближайших.

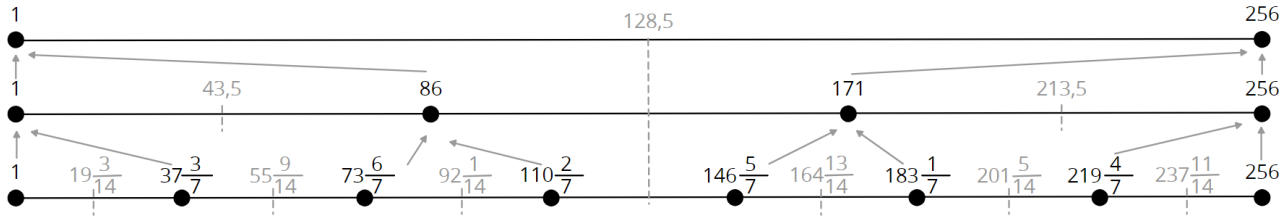
Алгоритм последовательно выполняется для значений i от 0 до 7 включительно.

Требуется узнать, сколько чисел 1 станет в матрице после выполнения алгоритма. В ответ запишите целое число.

Ответ: 55

Решение:

Рассмотрим на числовом отрезке последовательно несколько итераций для какого-либо числа, как оно будет меняться:



Черным отмечены точки, на которые будут заменяться наши числа. Серым отмечены точки – середины всех отрезков, с помощью которых и определяется какое число из двух соседей конкретно нужно выбрать (т.е. сначала смотрим, в какой отрезок попадает число, потом сравниваем это число с серединой данного отрезка, если наше число меньше середины отрезка, то на данной итерации число заменяется на число, равное левому концу данного отрезка, и наоборот).

Заметим, что после первой итерации все числа в матрице будут принадлежать числам из отрезка [1, 2^8], выбираемым первым шагом алгоритма. И так далее с каждой итерацией числа в матрице числа будут принадлежать выбранным на первом шаге алгоритма числам. Так же заметим, что на каждой итерации выбираемых первым шагом алгоритма числе будет в два раза меньше. Тогда можно заметить, что каждому числу из текущей итерации будут соответствовать 2 числа из предыдущей итерации (отмечено на схеме стрелками).

Так же, нетрудно заметить, что после выполнения работы алгоритма, все числа в матрице будут равны либо 1 либо 2^8. Тогда, чтобы узнать кол-во единиц в конце необязательно проделывать все шаги алгоритма, а достаточно узнать, сколько чисел из исходной матрицы будут меньше 128,5 – середины отрезка [1, 2^8]. В нашем случае таких числе будет 55.

9. Алгоритмизация. Анализ блок-схемы (3 балла) [Преобразование массива]

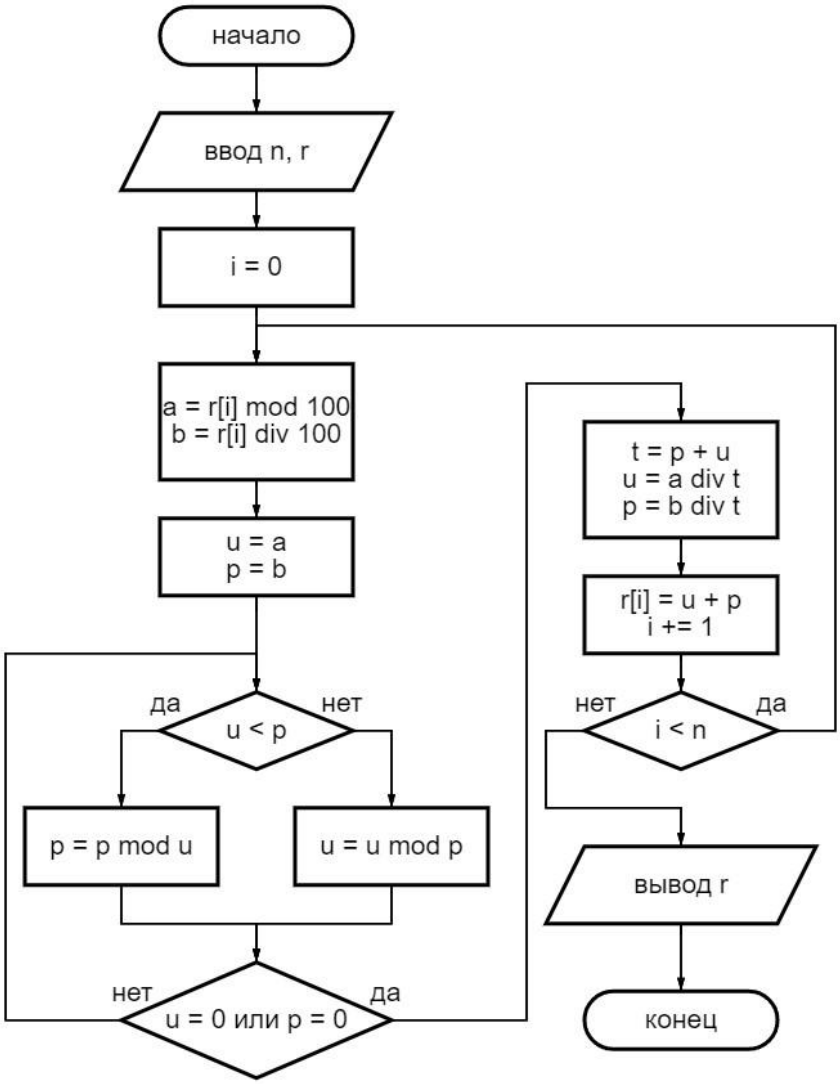
Вам дана блок-схема алгоритма, который получает на вход: число *n* – число элементов в массиве *r* и сам массив *r*. После алгоритм делает преобразования над элементами массива и возвращает изменённый массив *r*. Элементы в массиве нумеруются с нуля.

Алгоритм запустили для следующего массива (n = 5):

r = {7042, 2412, 981, 19590, 199}

В ответ запишите значения элементов массива после преобразований алгоритма подряд без пробелов в порядке их расположения в массиве.

(Допустим, что массив из двух элементов после работы алгоритма выглядит как: {44, 6}, тогда в ответ следует записать: 446)



Примечание:

1. “a += b” – переменной *a* присваивается значение суммы переменных *a* и *b*.
2. “p mod q” – взятие числа *p* по модулю *q* (остаток от целочисленного деления *p* на *q*).
3. “p div q” – целочисленное деление числа *p* на *q*.
4. “r[i]” – обращение к элементу массива под номером *i*.

Ответ: 831019100

Решение:

Проанализируем блок-схему: видим, что алгоритм перебирает все элементы массива в цикле, где переменная *i* хранит номер рассматриваемого элемента на текущей итерации цикла. В переменную *a* записывается число из двух последних цифр рассматриваемого элемента массива, а в переменную *b* – число из оставшихся цифр рассматриваемого элемента массива. Можно увидеть, что затем используется алгоритм Евклида для нахождения наибольшего общего делителя (НОД) чисел *a* и *b*. Рассчитанное значение НОД записывается в переменную *t*, после этого изначальные части, рассматриваемого элемента массива, *a* и *b* делятся на их НОД. Полученные значения суммируются и записываются в массив вместо старого значения.

r = {7042, 2412, 981, 19590, 199}

7042 => a = 42, b = 70 => НОД = 14 => r[0] = 3 + 5 = 8
2412 => a = 12, b = 24 => НОД = 12 => r[1] = 1 + 2 = 3
981 => a = 81, b = 9 => НОД = 9 => r[2] = 9 + 1 = 10
19590 => a = 90, b = 195 => НОД = 15 => r[3] = 6 + 13 = 19
199 => a = 99, b = 1 => НОД = 1 => r[4] = 99 + 1 = 100

Получили, что ответ равен: 831019100

10. Сортировка и фильтрация данных (2 балла)
[Сортировка Шелла]

Оля реализовала новую для себя сортировку. На вход программа получает массив из N элементов и набор значений d, расположенных от больших к меньшим. На каждом шаге алгоритма между собой сравниваются и сортируются все элементы, находящиеся друг от друга на расстоянии d, после чего берётся следующее значение для d и так продолжается до тех пор, пока d не будет равен 1 (см. пример). После этого массив будет гарантированно отсортирован.

Примечание:

Элемент A[j] находится на расстоянии d от элемента A[i], если A[i + d] = A[j] или A[i - d] = A[j].

D – массив из элементов d_i . Алгоритм выполняется для всех d_i из массива D последовательно.

Пример работы сортировки: для массива A = [7, 2, 4, 6] и значений D = [3, 2, 1]

d = 3: [7, 6]; [2]; [4] => [6, 7]; [2]; [4] => [6, 2, 4, 7]

d = 2: [6, 4]; [2; 7] => [4; 6]; [2; 7] => [4, 2, 6, 7]

d = 1: [4, 2, 6, 7] => [2, 4, 6, 7]

Получили [2, 4, 6, 7]

На вход программе подаётся:

Массив A = [72, 26, 114, 15, 21, 39, 6, 40, 115, 13, 7]

И значения D = [7, 3, 1]

Представим, что из-за сбоя алгоритм не закончил сортировку и остановился сразу перед значением d = 1 (выполнив сортировку для d = 7 и d = 3, но не выполнив для d = 1).

Какой элемент будет находиться в массиве по индексу “6” после этих преобразований? Индексы в массиве пронумерованы с нуля. В ответ запишите число.

Ответ: 40

Решение

Для того, чтобы узнать какой элемент окажется на 6-ой позиции в массиве, нам не обязательно выполнять сортировку полностью, а достаточно лишь рассмотреть элементы из которых выбирается значение A[6] на шаге d = 3. Это элементы с индексами: 0, 3, 6, 9.

0	1	2	3	4	5	6	7	8	9	10
72	26	114	15	21	39	6	40	115	13	7

Теперь рассмотрим какие из интересующих нас элементов менялись на шаге алгоритма для d = 7.

0	1	2	3	4	5	6	7	8	9	10
72	26	114	15	21	39	6	40	115	13	7

Интересующие нас группы: [72, 40]; [114, 13]; [15; 7] => [40, 72]; [13; 114]; [7; 15]

d = 7

0	1	2	3	4	5	6	7	8	9	10
40	26	13	7	21	39	6	72	115	114	15

Теперь выбираем элементы с индексами: 0, 3, 6, 9. Это [40; 7; 6; 114] => [6; 7; 40; 114]. Получается, что A[6] = 40

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

1. Теоретические основы информатики (1 балл)
[Базы данных]

Выберите из вариантов ниже название СУБД **не** с открытым исходным кодом:

1) Redis
2) PostgreSQL
3) Oracle
4) ClickHouse

Ответ: 3

2. Системы счисления (2 балла)
[Что-то со связью]

Ваня получил от друга сообщение по секретному каналу, однако качество соединения было крайне нестабильное, из-за чего часть сообщения пропала (пропущенные символы отмечены ‘_’). Вот, что получил Ваня от своего друга:

$\_2\_0u\_t_4 = 1xy5\_{\_8} = z\_wB_{16}$

Ваня понял, что это одно и то же число, записанное в различных системах счисления (нижний индекс, стоящий справа от числа у каждой из записей числа показывает систему счисления, в которой оно записано) через знак равно, и вместо x, y, z, w, u, t были какие-то цифры. Помогите Ване понять, какое число передал ему друг. В ответ запишите это число *в десятичной системе счисления*, если число нельзя определить однозначно, то укажите наибольшее. Если такого числа не существует, в ответе укажите NULL.

Ответ: 6955

3. Системы счисления. Основы логики (3 балла)

[Число A]

Петя, Вася и Таня на перемене между уроками придумывали числа и переводили их в разные системы счисления. Загадав некоторое число a, ребята попросили учителя отгадать его. Каждый из тройки сказал про число два факта.

Петя сказал:

- В десятичной системе счисления число a_{10} больше 64_{10}, но меньше 100_{10}.
- В четверичной системе счисления число a_{10} имеет в записи ровно два нуля.

Вася сказал:

- В четверичной системе счисления число a_{10} имеет в записи 4 цифры.
- В двоичной системе счисления в записи числа a_{10} есть две стоящие рядом единицы.

Таня сказала:

- В десятичной системе счисления число a_{10} не делится на 3_{10}.
- В десятичной системе счисления число a_{10} состоит только из четных цифр.

Учитель знает, что Петя всегда говорит правду, Таня всегда лжет, а Вася один раз говорит правду и один раз лжет. Помогите учителю найти загаданное ребятами число.

В ответ укажите искомое число *в десятичной системе счисления*.

Ответ: 72

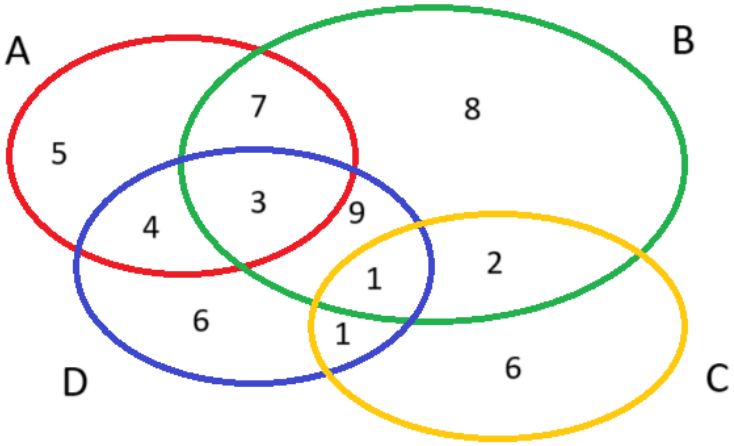
4. Основы логики (2 балл)

[Диаграмма Эйлера]

Вам дана диаграмма Эйлера и несколько логических выражений, в которых некоторые переменные неизвестны.

Выражения:

1. (A И B) ИЛИ (***X*** И (***Y*** ИЛИ НЕ D))
2. (A ИЛИ D) И (B ИЛИ ***Z***)



Для каждого числа на рисунке может быть истинно или ложно каждое из следующих утверждений:

- A = «Число внутри красного эллипса»
- B = «Число внутри зеленого эллипса»
- C = «Число внутри желтого эллипса»
- D = «Число внутри синего эллипса»
- A ИЛИ B = «Число внутри либо красного, либо зеленого эллипса»
- A И B = «Число внутри и красного, и зеленого эллипсов»
- НЕ A = «Число НЕ внутри красного эллипса»
- A ИЛИ (B И C) = «Число внутри красного эллипса или внутри зеленого и желтого эллипсов»

Подставьте вместо неизвестных переменных значения A, B, C или D так, чтобы при их подстановке в выражения в результате вычисления каждого из них получилась максимально возможная сумма входящих в него чисел. В ответ укажите последовательно значения A, B, C или D через запятую без пробелов в соответствии с их порядком в выражениях (то есть значения X, Y, Z соответственно).

Пример записи ответа: A,B,C

Ответ: D,D,D

5. Кодирование. Шифрование (3 балла)

[Послание от Почтальона Печкина]

Когда-то давно Почтальон Печкин решил зашифровать некоторое послание, состоящее из одного слова. Для этого он придумал некоторый алгоритм шифрования, и составил таблицу, в которой каждой букве сопоставлялся некоторый уникальный номер. Чтобы сохранить свое послание, Печкин записал его в таблицу размером 3x4 так, чтобы в каждой ячейке таблицы была ровно одна буква, и чтобы слово читать слева-направо и сверху-вниз. А потом заменил все буквы в таблице на соответствующие им цифры и применил к этой таблице свой алгоритм шифрования. Обе таблицы (с сопоставлением номеров и букв и с зашифрованным словом) Печкин поместил в две разные бутылки и выкинул в речку.

Очень много времени спустя отдыхая на речке кот Матроскин выловил бутылку с запиской, на которой была нарисована следующая таблица с цифрами:

18	13	18	5
14	29	8	17
14	8	19	2

Пройдя чуть дальше, у берега он нашел еще одну бутылку со следующей таблицей:

и	р	в	з	н	ш	б	к	е
1	2	3	4	5	6	7	8	9

Кот Матроскин догадался, что на первой табличке каким-то образом зашифрованы буквы из второй таблицы. Проведя исследования, он понял, что в каждой ячейке первой таблицы записана какая-то сумма, а конкретно у него есть три варианта:

1) сумма соседних по стороне клеток, включая значение самой клетки

Например: допустим дана следующая таблица сопоставления букв и номеров

а	к	н	о
1	2	3	4

Тогда построение таблицы 2х2 для шифрования слово ОКНА данным алгоритмом будет выглядеть следующим образом:

о(4)	к(2)	->	4+2 +3	2+4 +1	->	9	7
н(3)	а(1)		3+4+1	1+3 +2		8	6

2) сумма соседних клеток и по стороне, и по углу, не включая значение самой клетки

Например: допустим дана следующая таблица сопоставления букв и номеров

а	к	н	о
1	2	3	4

Тогда построение таблицы 2х2 для шифрования слово ОКНА данным алгоритмом будет выглядеть следующим образом:

о(4)	к(2)	->	2+3 +1	4+1 +3	->	6	8
н(3)	а(1)		4+1+2	3+2 +4		7	9

3) сумма соседних клеток только по стороне, не включая значение самой клетки

Например: допустим дана следующая таблица сопоставления букв и номеров

а	к	н	о
1	2	3	4

Тогда построение таблицы 2х2 для шифрования слово ОКНА данным алгоритмом будет выглядеть следующим образом:

о(4)	к(2)	->	2+3	4+1	->	5	5
н(3)	а(1)		4+1	3+2		5	5

Помогите коту Матроскину понять, какой же способ шифрования использовался для данной таблицы.

В ответ запишите зашифрованное слово, если известно, что оно было зашифровано в следующем порядке обхода таблицы: слева-направо, сверху-вниз, т.е. если после расшифровывания получалась следующая таблица с буквами:

ш	п	а
р	г	а
л	к	а

то ответом будет слово: шпаргалка

Ответ: безвершинник

6. Кодирование. Шифрование (2 балла)
[Смещения]

Карина придумала свой собственный способ шифрования слов. Шифрование состоит в том, что над всеми буквами слова производится ряд смещений их на некоторой 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 3 дает букву ‘Г’. Алфавит считается зацикленным, т. е. после буквы ‘Я’ идут снова ‘А’, ‘Б’, ‘В’ и так далее. Таким образом смещение ‘Я’ на 3 даст букву ‘В’.

Карине очень нравится число 18, поэтому она решила использовать его в своем шифре, а именно – сдвигать все буквы последовательно на все делители числа 18, и сделать 18181818 смещений для каждой буквы. Т. е. сначала над всеми буквами будет проведена операция смещения на 1, потом – опять же, над всеми буквами – смещение на 2, потом – на 3, потом – на 6, и так далее, пока не будет выполнено смещение на 18. Далее выполнение смещений начинается сначала – смещение на 1, на 2 – и так далее. В итоге для каждой буквы должно быть выполнено 18181818 смещений.

Помогите Карине зашифровать слово «ПРИВЕТ» ее методом, в ответе укажите зашифрованное сообщение заглавными буквами.

Примечание. Русский алфавит: А, Б, В, Г, Д, Е, Ё, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я
Пример шифрования слова ‘Ёж’ в случае, если нужно выполнить всего 2 операции смещения для каждой из букв: первое смещение производится на 1 – получаем слово ЖЗ. Второе смещение производится на 2 – получаем слово ИЙ. 2 операции смещения выполнены, шифрование завершено.

Ответ: ХЦОЗКШ

7. Кодирование. Графика (3 балла)
[Композиция]

Витя любит составлять картинки из геометрических фигур. Он нашел два способа их сохранения:

1) Растровый вариант:

В этом случае для кодирования изображения последовательно перечисляются цвета пикселей. Так как Витя рисует картинки в оттенках серого, на запись цвета каждого пикселя отводится 1 байт. Этот байт содержит в себе информацию об оттенке серого. Например, 0 – это черный цвет, 255 – белый, а остальные числа отражают возможные промежуточные значения яркости. Границы изображения определяется по крайним точкам фигур на рисунке.

2) Векторный вариант:

Геометрические фигуры кодируются особым способом. Первый байт всегда указывает на тип фигуры, а за ним следуют байты задающие параметры фигуры. Этот вариант хранения изображений позволяет рисовать два типа фигур:

- **Прямоугольник** 1|X|Y|W|H|C

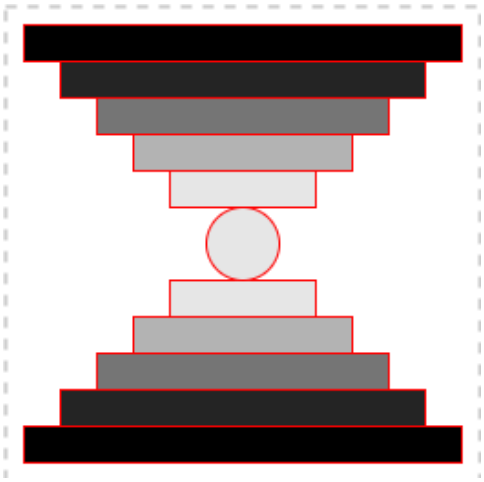
Запись	**Описание**	**Размер (байт)**
1	Тип фигуры	1
X	Координата x левого верхнего угла	4
Y	Координата y левого верхнего угла	4
W	Ширина	4
H	Высота	4
C	Цвет фигуры	1

- **Круг** 2|X|Y|R|C

Запись	**Описание**	**Размер (байт)**
2	Тип фигуры	1
X	Координата x центра круга	4
Y	Координата y центра круга	4
R	Радиус	4
C	Цвет фигуры	1

Все размеры и координаты задаются в виде целого числа пикселей. Возможным переполнением переменных можно пренебречь.

Витя задумал нарисовать квадратное изображение (см. рисунок ниже) и ему стало интересно при каком *минимальном* размере картинки растровое изображение будет занимать больше байт чем векторное. Красным цветом показаны границы фигур для видимости, на картинке Вити их не будет.



В ответе укажите целое число пикселей – минимальную *длину* стороны квадратной картинки, при которой растровое изображение будет занимать больше байт, чем векторное.

Ответ: 14

8. Теория игр (2 балла)

[Распределение задач]

Два программиста создают перспективный стартап. У них есть список задач на месяц, а чтобы распределить их между собой, они придумали следующую игру:

Пусть, изначально в списке находится n задач. Программисты по очереди забирают задачи из списка и записывают их в свой трекер задач. Тот программист, на чьём ходу список незанятых задач закончится – победит.

В свой ход каждый из программистов может взять себе либо три задачи (или все оставшиеся, если задач в списке меньше трёх) либо взять ровно столько задач, чтобы в списке осталось m>>1 задач, где m – число задач в списке на начало хода этого программиста, а >> - операция битового сдвига на 1 вправо, то есть самый младший разряд числа в двоичной системе счисления отбрасывается, а все остальные разряды числа смещаются на один вправо. Пример операции 5 >> 1: представим 5 в двоичной системе счисления: 101, 101 >> 1 = 10, переведём 10 в десятичную систему и получим цифру 2.

Какое *минимальное* число задач в диапазоне от 20 до 27 включительно может быть в списке изначально, чтобы гарантированно победил программист, делающий ход первым?

Ответ: 23

9. Комбинаторика (1 балл)

[Магические числа]

Лера и Даша играли с числами, перебирая комбинации цифр в шестизначных числах. Даша предложила задать для чисел правила и назвать такие числа магическими.

Число считается магическим, если соблюдаются правила:

1. Цифры в числе стоят в порядке возрастания
2. В числе не более трех четных цифр
3. Сумма цифр числа - нечетное число
4. Все цифры числа - различны

Помогите Лере и Даше найти *количество* шестизначных чисел, которые являются магическими.

В ответе укажите количество таких чисел.

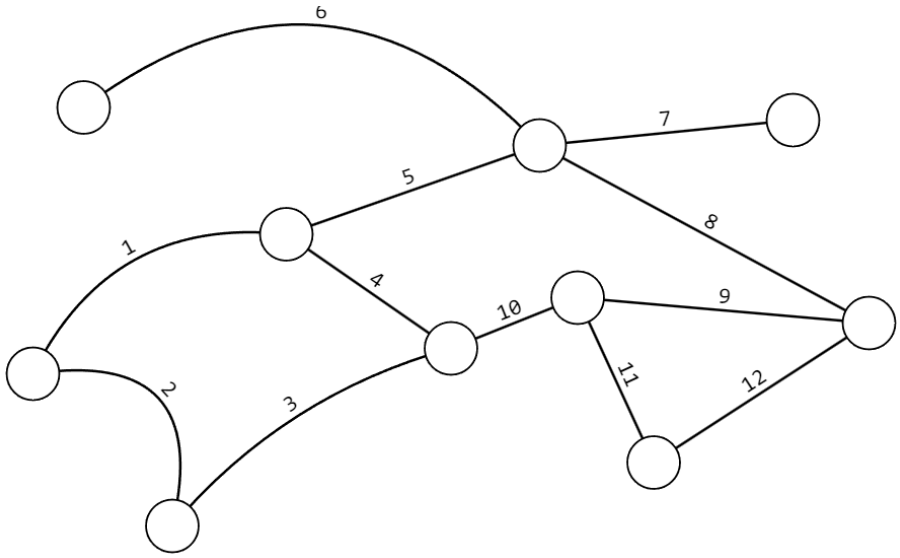
Ответ: 44

10. Моделирование на графе (2 балла)

[Стратегическая перестройка]

Василий играет в компьютерную стратегию о развитии своей империи в эпоху средневековья. Василий может смотреть все свои города на карте, где они схематично представлены в виде графа, вершины которого – это города, а рёбра графа – пронумерованные дороги между городами. Неожиданно, Василий столкнулся с нехваткой ресурсов на строительство, поэтому он решает разобрать несколько дорог и получить затраченные на них средства обратно. Помогите Василию выбрать, какие дороги можно убрать так, чтобы при этом сохранилась возможность из любого города добраться по дорогам до любого другого, а также сумма полученных ресурсов с разобранных дорог была максимальной?

В ответ запишите *номера дорог в порядке возрастания без пробелов*.



Номер	1	2	3	4	5	6	7	8	9	10	11	12
Ресурсы	3	9	5	4	21	22	30	14	22	13	24	23

Ответ: 2911

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

1. Архитектура компьютера (1 балл)
[Облачные вычисления]

Выберите верное утверждение для архитектуры облачных вычислений

1) Контроль над выбором комплектующих для серверов лежит на пользователе
2) Сервера используются только их владельцем
3) Для оплаты ресурсов используется модель pay-as-you-use (оплата по мере использования ресурсов)
4) Пользователь самостоятельно чинит сломанные комплектующие серверов

В ответ укажите одно целое число – номер верного утверждения

Ответ: 3

2. Архитектура компьютера (1 балл)
[CI/CD]

Выберите среди перечисленного инструменты, которые предназначены для автоматизации процессов сборки, тестирования и доставки ПО в рамках методологии Continuous Integration/Continuous Deployment (CI/CD)?

1. Jenkins
2. DBeaver
3. Gitlab CI
4. VS Code
5. Allure

В ответе укажите номера выбранных вариантов ответа через пробел. Пример записи ответа: «1 2 3 4».

Ответ: 1 3

3. Моделирование (2 балла)
[Байдарки]

Почтальон Печкин и кот Матроскин решили участвовать в соревновании по гребле на байдарках. Но при подготовке они смогли найти только одну двухместную байдарку и только одно весло. Тогда Печкин и Матроскин решили плыть вместе и в случае победы – разделить выигрыш пополам. Но возник вопрос, как они будут управлять вдвоем байдаркой с помощью одного весла. Было принято решение грести по очереди. Они договорились о том, что каждый во время своей очереди должен прогрести целое число метров – от 1 до 10. Первым решил грести почтальон Печкин.

Для того, чтобы дистанция считалась пройденной, им нужно проплыть 24 метра. Сколько у Матроскина и Печкина есть различных способов разделения между собой дистанции?

Примечание:

Примеры способов разделения для дистанции в 11 метров

1) Печкин – 1 метр => Матроскин – 10 метров

2) Печкин – 3 метра => Матроскин – 6 метров => Печкин – 2 метра

и т.д.

Ответ: 8327186

4. Моделирование (2 балл)

[Ёлка]

Новый год уже давно прошел, а Аня так и не разобрала украшенную ёлку.

Ёлка украшена шариками четырёх цветов: красными, золотистыми, зелеными и белыми. Условно ёлку по высоте можно разделить на несколько уровней: на самом верхнем, будем считать его первым, висит одна игрушка, на втором – чуть ниже первого – две игрушки, на третьем – три, и так далее. Известно, что шары вешали на ёлку строго в порядке следования цветов: красный, золотистый, зеленый, белый, снова красный, золотистый, зеленый, белый и так далее, слева-направо, сверху вниз.

Пример украшения ёлки по этому правилу для 4-уровневой ёлки:

Известно, что на Аниной ёлке 52 уровня игрушек.

Анин кот, Василий, решил помочь хозяйке освободить ёлку от украшений: он решил каждый день сбрасывать с ёлки по одной или несколько игрушек. Сбрасывать случайные игрушки ему было неинтересно, поэтому он выработал для себя следующее правило: за один день нужно сбросить с одного из уровней все игрушки одного цвета и по одной игрушке каждого из оставшихся на этом уровне цветов. Василий знает, что белые шары – Анины любимые, поэтому он хочет действовать так, чтобы на ёлке как можно раньше остались только белые шары, и чтобы к моменту, когда не останется шаров других цветов, белых шаров осталось как можно больше.

К концу какого дня помощи кота на ёлке останутся только белые шары, и на скольких уровнях ёлки к этому моменту останется хотя бы по одной игрушке?

В ответ укажите два целых числа без пробела – первый по счету возможный день, к вечеру которого на ёлке останутся только белые шары, и число уровней, на которых к этому моменту останется хотя бы по одной игрушке. Например, если ответ «к концу 123 дня; 45 уровней», то в ответ необходимо записать «12345».

Ответ: 13941

5. Моделирование (2 балла)

[Строка]

Вам дана исходная строка: “software developer”. Каждый буквенный символ строки изменяется по следующему алгоритму:

1. Вычисляется его порядковый номер в алфавите (буквы в алфавите нумеруются с 1).
2. Найденный порядковый номер умножается на 2.
3. Если полученное число больше 26, то из него вычитается 26.
4. Вместо исходной буквы записывается буква с вычисленным порядковым номером.

Для каждого буквенного символа строки алгоритм был выполнен 2024 раза. Необходимо определить сколько уникальных буквенных символов получилось в итоговой строке.

В ответ необходимо указать количество уникальных буквенных символов.

Ответ: 10

6. Моделирование (2 балла)

[Метрополитен]

Абитуриентка Маша поступила в университет и приехала в город Мегаполис. Удобнее всего Маше передвигаться по городу на метро, поэтому она взялась изучать метрополитен города, в частности путь от станции возле общежития до станции возле университета.

Начальная станция метро - Н

Конечная станция метро - З

Пересадка с одной станции метро на другую у Маши занимает 5 минут, пересадка между станциями производится в том случае, если нужно изменить цвет ветки метро, по которой едешь.

На картинке представлена карта метрополитена. Одна ветка метро – это путь, по которому поезд едет без пересадок. Одна ветка обозначена одним цветом.

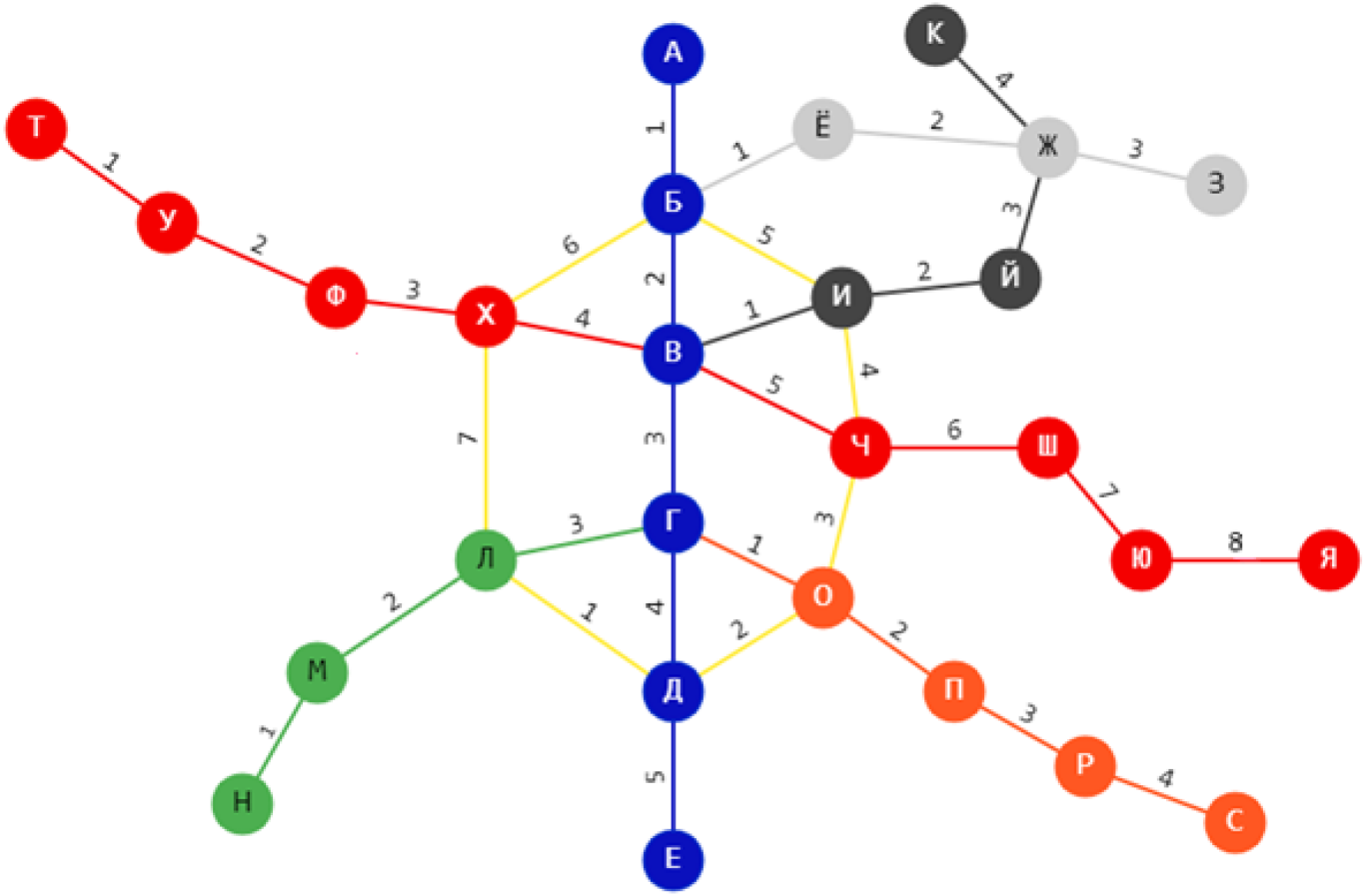
На пересечении веток метрополитена можно пересаживаться с одной ветки на другую, каждая пересадка занимает 5 минут. Если поезд метро останавливается на пересадочной станции, но пассажир не переходит в поезд, едущий по ветке другого цвета, то это не считается пересадкой.

Цифры, указанные над дорогами между станциями, это время, за которое поезд проезжает этот путь.

Например:

Путь от станции Ф до станции Б занимает 14 минут.

ФХВБ = ФХ (3 минуты) + ХВ (4 минуты) + Пересадка с красной ветки на синюю (5 минут) + ВБ (2 минуты)



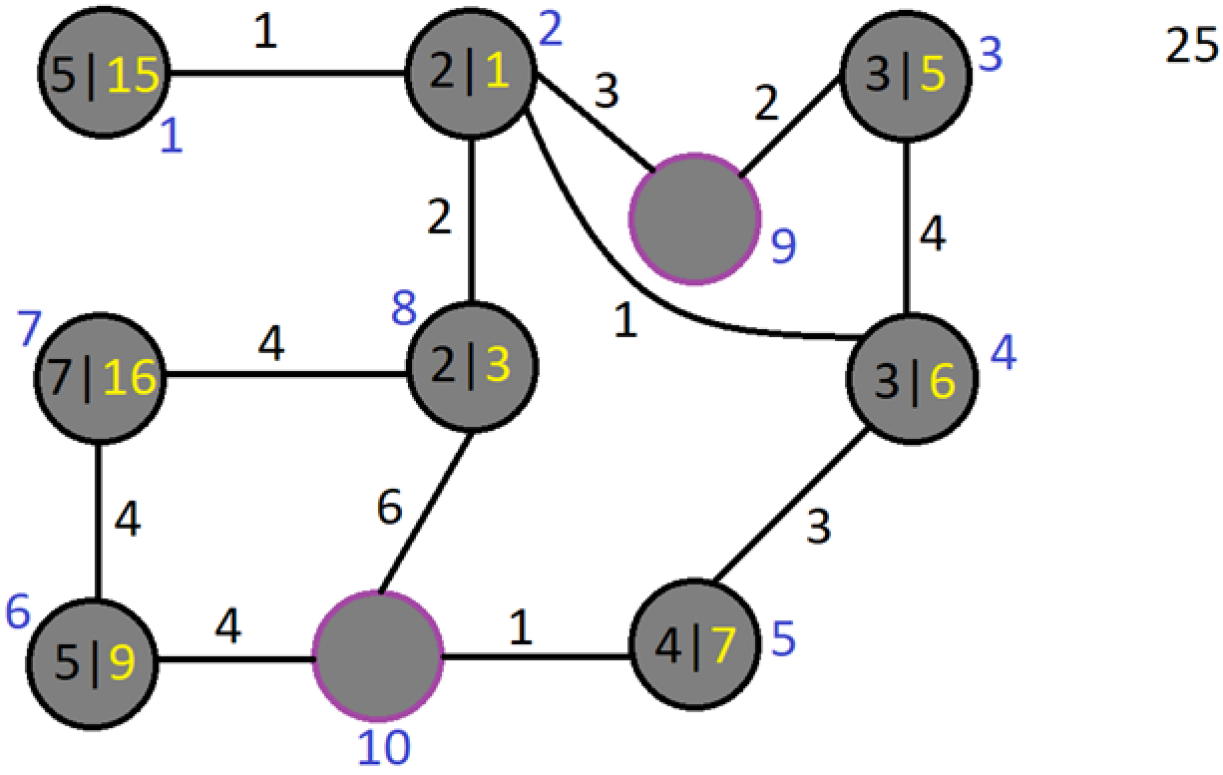
Помоги Маше определить путь, по которому она может добраться из станции Н в станцию З быстрее всего.

В ответе введи станции подряд без пробелов, включая начальную и конечную станции.

Пример ввода ответа если бы начальной станцией была А, а конечной Г: АБВГ
Ответ: НМЛГВБЁЖЗ

7. Моделирование (3 балла)
[Подземелья и гоблины]

Ваня очень любит играть в компьютерную игру «Подземелья и гоблины». В игре персонаж спускается в одну из комнат подземелья. Подземелье состоит из множества комнат, которые соединены друг с другом тоннелями, но необязательно каждая с каждой. Проход по каждому тоннелю требует определённое количество времени. В каждой комнате есть монеты, которые охраняет гоблин. Как только игрок входит в комнату (в том числе, если входит в подземелье через комнату), то гоблин нападает на него. Чтобы одолеть гоблина в комнате нужно затратить определённое количество времени. Как только гоблин повержен, игрок забирает монеты в комнате. Если игрок уже был в комнате, то сражаться с гоблином не нужно. По пути в подземелье Ваня нашёл карту, на которой была изображена схема подземелья: черными кружками отмечены комнаты, линиями, соединяющими кружки, отмечены тоннели между комнатами, фиолетовыми кружками отмечены комнаты с выходом из подземелья, черными числами внутри кружков отмечено время битвы с гоблином, жёлтыми числами внутри кружков отмечено количество золота в комнате, чёрными цифрами рядом с чёрными линиями отмечено время пути по тоннелю, синим числом рядом с кружком отмечен номер комнаты. Также в углу карты было отмечено, сколько времени у Вани есть, чтобы зайти в подземелье и выйти из него до момента его обрушения. Как только Ваня подошёл к подземелью, он заметил, что может войти в него через любую комнату, а выйти только через комнаты с выходом. Ваня очень ценит своё время и перед спуском решил разработать стратегию обхода комнат, чтобы собрать максимальное количество монет и успеть выйти из подземелья до его обрушения. Напишите в ответ последовательность комнат, которую должен обойти Ваня, чтобы получить максимальное количество монет, и, через пробел, количество монет после выхода из подземелья.



Пример ответа: 129 16

Ответ: 82124510 32

8. Программирование и алгоритмизация (3 балла)

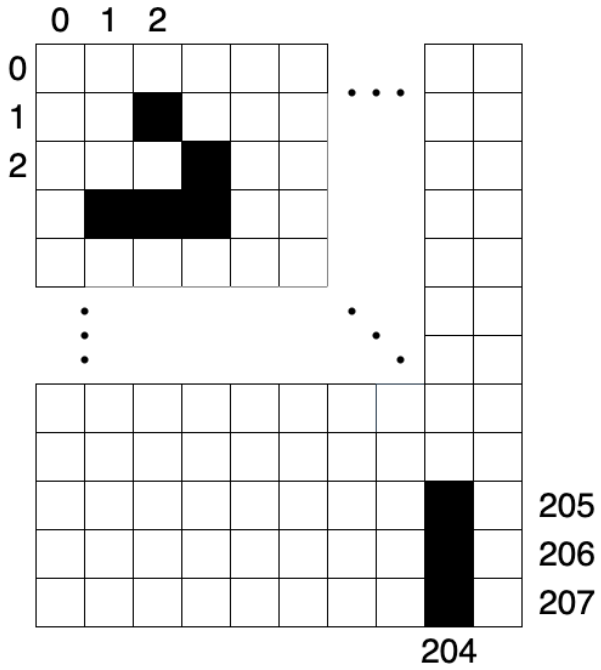
[Игра «Жизнь»]

В 1970 году Джон Конвей придумал игру под названием “Жизнь”. Она состоит буквально из 4 правил:

- Место действия игры — размеченная на клетки неограниченная плоскость
- Каждая клетка имеет 8 соседей (сверху, снизу, слева, справа и по диагоналям) и может находиться в двух состояниях: быть живой или мертвой
- В пустой (мертвой) клетке, с которой соседствуют ровно 3 живые, зарождается жизнь
- Если у живой клетки ровно 2 или ровно 3 живых соседа, то она продолжает жить, иначе погибает

На поле выставляется стартовая конфигурация, а затем ко всем клеткам одновременно применяются описанные выше правила. Один раз применив правила к каждой клетке (выполнив одну итерацию), мы получаем новое поколение.

Перед началом игры поле имеет следующую конфигурацию (см. рисунок). Координаты отсчитываются с левого верхнего угла, начиная с поля (0,0). Если ячейка окрашена черным, то она живая, если белым, то нет.



Необходимо выяснить, через сколько итераций на поле не останется ни одной живой клетки.

Примечание

В ответе ожидается целый номер самой ранней итерации, на которой исчезли все живые клетки или -1 если такая ситуация недостижима

Ответ: 808

9. Программирование и алгоритмизация (3 балла) [Обработка изображения]

Яна работает над программой по сжатию чёрно-белых изображений.

Чёрно-белая картинка представлена в программе в виде двумерного массива *a* размером *N* на *M*, где *N* – число строк в массиве, а *M* – число столбцов. Значение каждого элемента двумерного массива находится в диапазоне от 0 до 255.

Яна узнала, что если применить к каждой строке исходного массива одну из двух-функций фильтров - f1 или f2, то можно добиться более эффективного сжатия изображения.

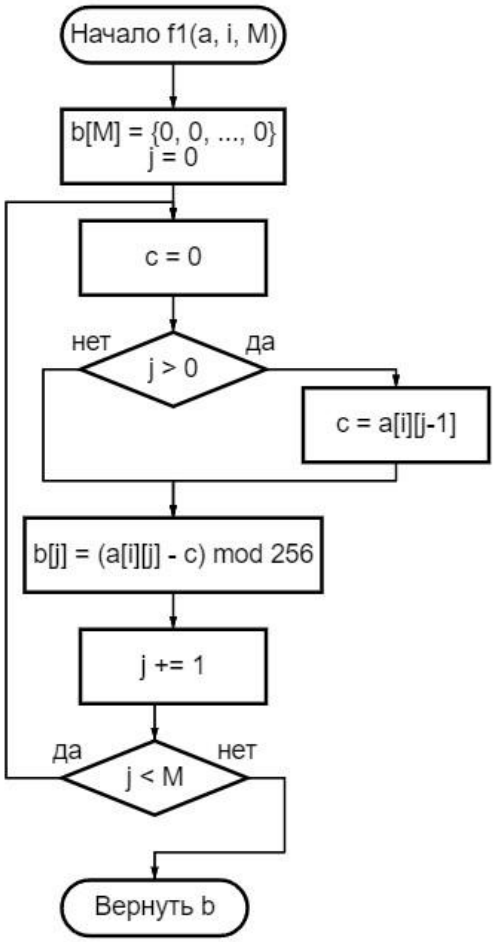
Яна выбирает и применяет для каждой строки из *a* тот фильтр, при котором в новой строке будет меньше уникальных элементов (при равном числе уникальных элементов в строках, полученных после работы фильтров, выбирается фильтр f1). Применение любого из фильтров не меняет исходный массив *a*, вместо этого новые строки записываются в новый массив *c*.

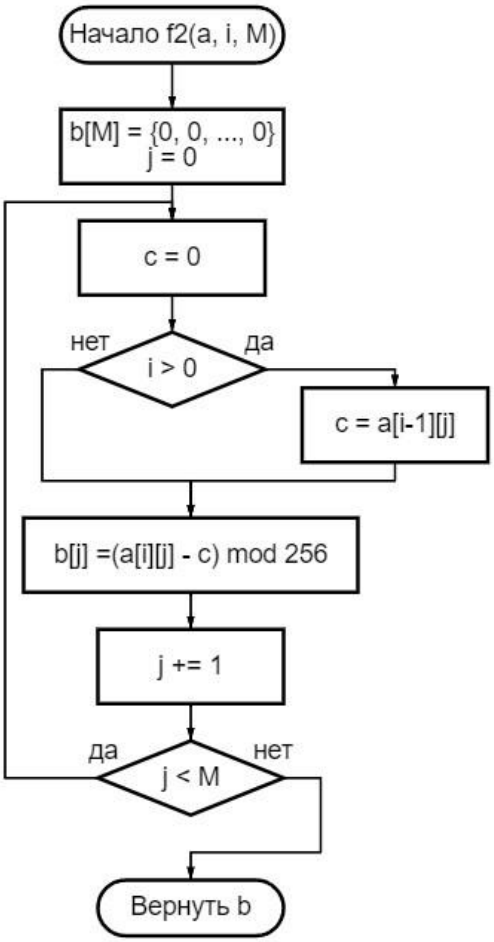
Массив, с которым работает Яна, выглядит следующим образом (*N*,*M* = 4):

20	128	236	17
42	84	135	177
62	101	152	194
60	0	200	4

У Вас есть блок-схемы, описывающие работу фильтров *f1, f2*. На вход каждому фильтру передаётся массив *a*, номер рассматриваемой строки *i* (строки нумеруются с 0), и число элементов в строке *M*. Определите, какой фильтр был выбран для кодирования каждой из строк исходного массива. В ответ напишите подряд номера выбранных фильтров без пробелов в порядке возрастания номера строки (сверху вниз).

Например, если Вы решили, что для изображения 2x2 для 0-й строки был применён 2-й фильтр, а для 1-й строки 1-й фильтр, то в ответ нужно записать: 21





Примечания:

1. Номера фильтров *f1* = 1, *f2* = 2.
2. “a += b” – переменной *a* присваивается значение суммы переменных *a* и *b*
3. “arr[n] = {0, 0, … , 0}” – массив из *n* элементов, заполненный нулями.
4. “p mod q” – взятие числа *p* по модулю *q*.

Если p < 0, то:

p mod q = q – (-p % q)

где % - остаток от целочисленного деления

Если p >= 0, то: p mod q = p % q

Ответ: 1121

10. Сортировка и фильтрация данных (2 балла) [Сортировка «спиралью»]

Вася придумал свой алгоритм сортировки двумерных массивов – сортировка «спиралью». Первые несколько итераций сортировки «спиралью» для массива nxm:

1. сортируем $1^{ую}$ строку от $1^{ого}$ до $m^{ого}$ значения (слева направо)
2. сортируем $m^{ый}$ столбец от $1^{ого}$ до $n^{ого}$ значения (сверху вниз)
3. сортируем $n^{ую}$ строку от $m^{ого}$ до $1^{ого}$ значения (справа налево)
4. сортируем $1^{ый}$ столбец от $m^{ого}$ до $2^{ого}$ значения (снизу вверх)
5. сортируем $2^{ую}$ строку от $1^{ого}$ до $(m-1)^{ого}$ значения (слева направо)
6. сортируем $(m-1)^{ый}$ столбец от $2^{ого}$ до $(n-1)^{ого}$ значения (сверху вниз)

и т.д. сортируя поочередно столбцы и строки идя по спирали

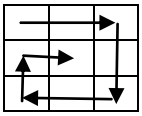
Иллюстрация применения сортировки «спиралью»:

0 шаг				1 шаг				2 шаг				3 шаг				4 шаг				5 шаг		
2	8	4		2	4	8		2	4	1		2	4	1		2	4	1		2	4	1
7	9	3		7	9	3		7	9	3		7	9	3		8	9	3		8	9	3
6	5	1		6	5	1		6	5	8		8	6	5		7	6	5		7	8	6

Нетрудно заметить, что за один проход (или эпоху) по всему массиву – массив не обязательно будет полностью отсортирован по «спирали». Массив, приведенный в примере, будет полностью отсортирован по «спирали» за две эпохи.
Для тестирования своего алгоритма Вася заполнил двумерный массив размера 4x4 следующими числами:

9	27	13	5
23	15	44	84
7	30	78	16
54	28	36	73

Эпохой сортировки называет один проход данной сортировки по всему массиву.
Сколько Васе понадобиться эпох, чтобы его массив стал полностью отсортирован по спирали? В ответ запишите одно целое число – количество эпох.
Проход массива по спирали имеет следующий вид:



Пример:
Изначальный массив:

3	*10*	*8*
17	*33*	*2*
5	*27*	*13*

Полностью отсортированный по спирали массив будет выглядеть следующим образом:

2	*3*	*5*
27	*33*	*8*
17	*13*	*10*

Ответ: 4