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:

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

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

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

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

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

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 $F(X,Y,Z) = Y \to X \wedge Z$

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”.

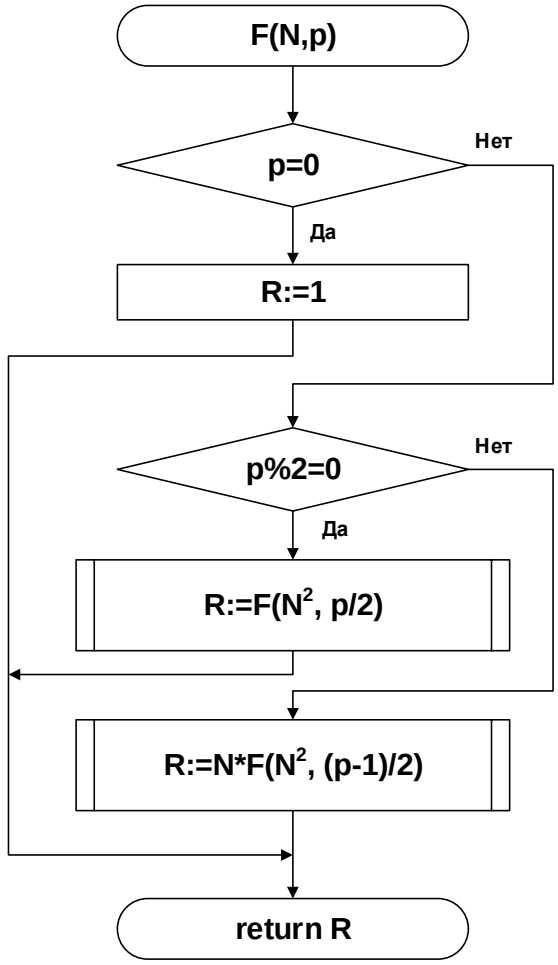
Аналогично поступим с остальными номерами символов и получим следующую таблицу:

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:

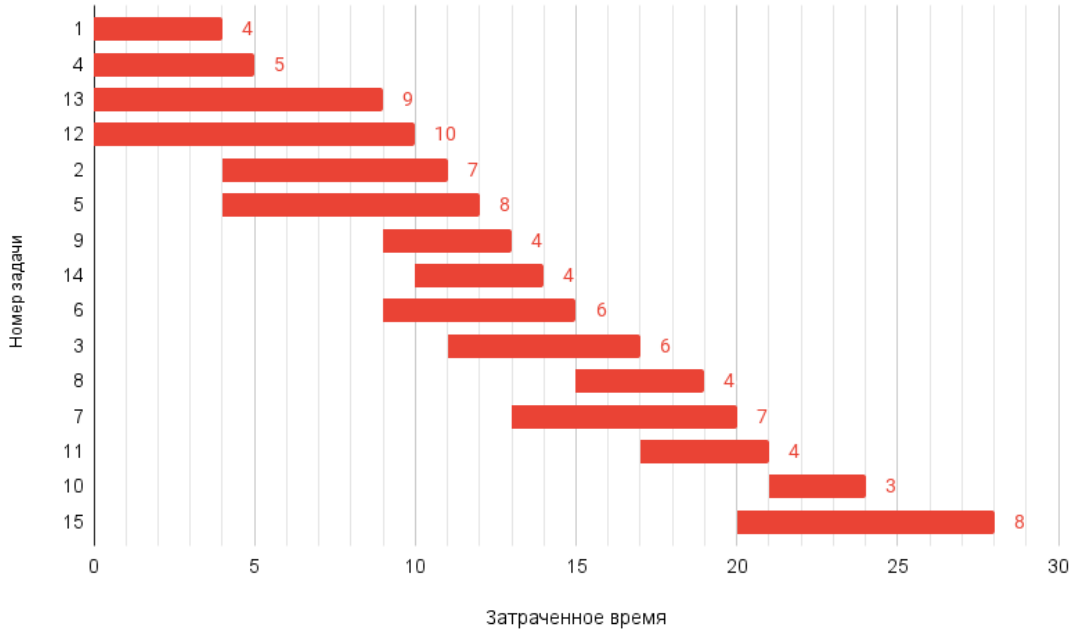


Диаграмма Ганта - это разновидность гистограммы, на которой показывается порядок и продолжительность выполнения поставленных задач. Каждая полоса на диаграмме показывает, когда начинается и заканчивается выполнение задачи. Диаграмма Ганта также показывает зависимые задачи, которые нужно выполнить перед 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:

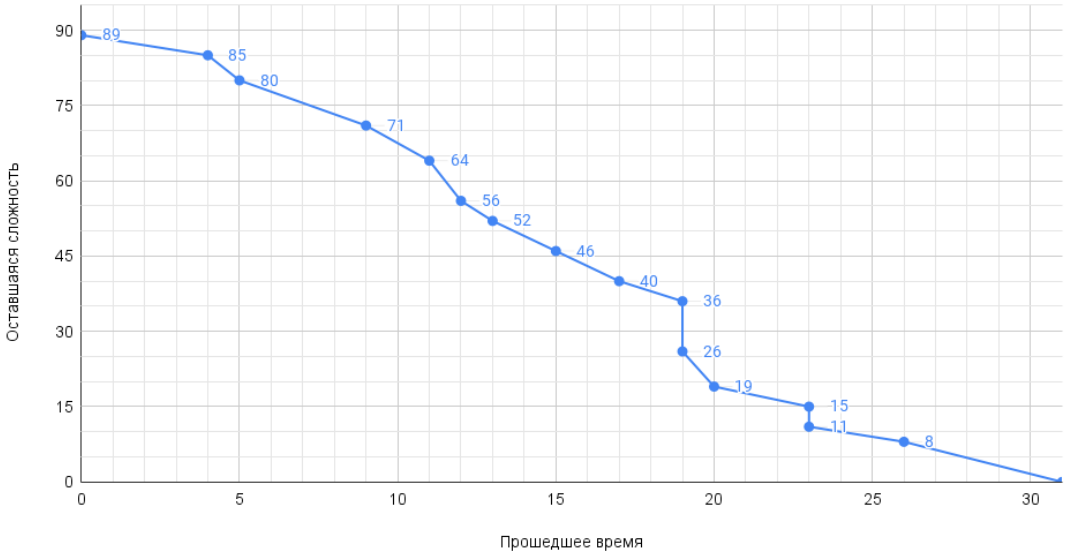


Диаграмма сгорания - это разновидность графика, на котором каждая точка ломаной на графике показывает момент изменения общей оставшейся сложности. Общая оставшаяся сложность считается как сумма сложностей всех оставшихся задач, в том числе выполняемых на текущий момент (без учёта степени их выполнения). 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	
	turn D
OK	
	move
OK	
	turn R
OK	
	move
OK	
	move
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	
	move
OK	
	move
OK	
	turn R
OK	
	camera
bbbb gbbb bbbb	
	move
OK	
	move
OK	
	turn D
OK	
	camera
bbbb bbbr bbbb	
	wait 10
OK	
	camera
bbbb bbbb bbbg	
	move
OK	
	move
OK	

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 $\min_{ai \in group0,\ aj \in group1} ai \oplus aj$,

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.