Индивидуальный предметный тур. Информатика

Для возрастной категории 7–8 классов были предложены задачи A–E. Для возрастной категории 9–11 классов были предложены задачи B–F.

Задача A. Украшение

Стандартный ввод. Стандартный вывод. 1 секунда. 256 мегабайт.

Ювелиру Артему девушка Анастасия должна принести несколько незамкнутых цепочек. Артем должен будет соединить все в единую незамкнутую цепь. Цепочки могут иметь разную длину. Длина цепочки – это количество звеньев в ней. Самая короткая цепь может состоять из одного звена.

Цепочки соединяют в единую незамкнутую цепь. Для этого нужно какие-то звенья распилить и соединить ими некоторые цепочки.

Помогите найти минимальное количество звеньев, которые надо распилить, чтобы затем с их помощью соединить все цепочки в единую цепь.

Даны четыре тестовых массива с длинами цепочек, для которых требуется найти ответ. Если вы не знаете ответ для теста, запишите *-1*.

Номер теста	Массив	Ответ
1	*3,5*	(поле ответа)
2	*4, 4, 4, 4, 4*	(поле ответа)
3	*2, 3, 5, 2, 4*	(поле ответа)
4	*2, 30, 1, 123, 4, 8, 10, 5, 12, 4, 3, 9*	(поле ответа)

Как ответ на эту задачу прикрепите код или ***.txt*** файл. Если отсылаете ***.txt*** файл, то на первой строчке должно быть записано целое число ответ для первого теста, во второй строчке должно быть записано целое число ответ для второго теста и тд. Если отсылаете код, то на первой строке выходных данных выведите ответ для первого теста, во второй строчке выведите ответ для второго теста и тд.

По этой задаче на проверку принимается не более трех файлов.

Каждый тест оценивается от 0 до 25 баллов по формуле

$$\left\lfloor 25 * \frac{1}{1 + 10 * \left| points_{participant} - points_{max} \right|} \right\rfloor$$

где $points_{participant}$ – ответ участника, $points_{max}$ – ответ жюри.

$\lfloor x \rfloor$ обозначает округление вниз, то есть наибольшее число, меньшее или равное x. Например, $\lfloor 2.7 \rfloor = 2$, $\lfloor \pi \rfloor = 3$ и $\lfloor 5 \rfloor = 5$.

Решение:

Заметим, что если среди исходных цепочек есть однозвенная цепочка (т.е. состоящая только из одного звена), то её нужно использовать для соединения двух других цепочек. Это оптимальная операция, т.к. в результате её выполнения количество несоединённых цепочек уменьшается сразу на 2.

Следовательно, сначала нужно использовать только цепочки наименьшей длины.

Идеи алгоритма:

1) сначала отсортировать все исходные цепочки по возрастанию их длины;

2) затем по очереди использовать самые короткие цепочки для соединения самых длинных цепочек.

Пример решения на языке программирования Python:

```
def solve():
    n = int(input())
    a = [int(i) for i in input().split()]

    a.sort()

    if n == 1:
        print(0)
        return

          ans = 0
          for i in a:
              for j in range(i):
                  if n <= ans + 1:
                      print(ans)
                      return

                  ans += 1
              n -= 1

          print(ans)
          return

      for _ in range(4):
          solve()
```

Задача B. Игра с часами

Стандартный ввод. Стандартный вывод. 1 секунда. 256 мегабайт.

Разработчики из VK придумали новую игру с часами.

На часах h часов и m минут. Игрок бесконечное количество раз заменяет в своём воображении текущее время на время, которое будет на часах через столько минут, сколько в текущем времени часов. Что в итоге получится?

Формат входных данных

Дано 2 целых числа h и m – текущее время ($0 \le h \le 23$, $0 \le m \le 59$).

Формат выходных данных

Выведите 2 числа h' и m' – итоговое время.

Система начисления баллов

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи	Информация о проверке
0	-	пример из условия		полная
1	20	$h = 23, m \geq 37$		первая ошибка
2	20	$h = 23$	1	первая ошибка
3	40	$h > 0$	0, 1, 2	первая ошибка
4	10	$h = 0$		первая ошибка
5	10	без дополнительных ограничений	0 – 4	первая ошибка

Пример

стандартный ввод	стандартный вывод
16 13	0 18

Пояснение к примеру

16:13 → 16:29 → 16:45 → 17:01 → 17:18 → 17:35 → 17:52 → 18:09 → 18:27 → 18:45 → 19:03 → 19:22 → 19:41 → 20:00 → 20:20 → 20:40 → 21:00 → 21:21 → 21:42 → 22:03 → 22:25 → 22:47 → 23:09 → 23:32 → 23:55 → 0:18

Напоминаем, что в часе 60 минут, а в сутках 24 часа.

Решение

В данной задаче достаточно аккуратно смоделировать описанный в условии задачи процесс.

Пример полного решения на Python 3:

```
h, m = map(int, input().split())
while h != 0:
    m += h
    if m >= 60:
        m -= 60
        h = (h + 1) % 24
print(h, m)
```

Задача C. Полоса загрузки

Стандартный ввод. Стандартный вывод. 1 секунда. 256 мегабайт.

Я ни разу не работал над игрой, в которой полоса загрузки была бы не фейковой.

Рауль Рубио Мунаррис,
директор Tequila Works

Часто неадекватно дёргается полоса загрузки? Хватит это терпеть! Давайте учтём все факторы и реализуем реалистичное отображение процента загрузки файлов в робота.

Изначально полоса ориентируется на процент количества загруженных файлов. Сделайте так, чтобы она ориентировалась на процент от суммарного времени загрузки файлов. В качестве ответа выведите максимальную разницу

в отображаемом проценте в старой и новой полосе, которую можно наблюдать в процессе загрузки.

Отображаемый процент всегда округляется вниз и меняется только в момент окончания загрузки файла. Для лучшего понимания прочтите пояснение к примеру.

Формат входных данных

В первой строке ввода содержится целое число n – количество файлов *(1 ≤ n ≤ 3*10^5)*.

Вторая строка содержит n положительных целых чисел t_i – время загрузки i-го файла ($\sum_{i=1}^{n} t_i \leq 10^{16}$).

Файлы даны в порядке их загрузки в робота. Менять порядок запрещено.

Формат выходных данных

Выведите максимальную разницу (модуль разности) в отображении при рассматривании процента загруженных файлов и процента от суммарного времени загрузки файлов.

Система начисления баллов

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи	Информация о проверке
0	-	пример из условия		полная
1	10	$n \leq 2$		первая ошибка
2	10	$n \leq 3$	1	первая ошибка
3	15	$n = 100$		первая ошибка
4	15	n кратно 100	3	первая ошибка
5	10	$t_i = t_j$ для всех i, j		первая ошибка
6	10	$n, t_i \leq 3000$	0	первая ошибка
7	10	$n \leq 3000$	0–3, 6	первая ошибка
8	20	без дополнительных условий	0–7	первая ошибка

Пример

стандартный ввод	стандартный вывод
5 1 4 2 5 3	14

Пояснение к примеру

Таблица для примера:

После загрузки файла номер	-	1	2	3	4	5
Процент файлов	0	20	40	60	80	100
Процент времени	0	6	33	46	80	100
Модуль разности	0	14	7	14	0	0

Решение

В данной задаче достаточно аккуратно смоделировать описанный в условии задачи процесс и найти максимальный модуль разности.

В случае рассмотрения отображаемых процентов в порядке возрастания количества загруженных файлов для быстрого расчёта суммы времен загрузки первых i файлов можно считать названную сумму не заново, а прибавить t_i к уже посчитанной ранее сумме времен загрузки первых $i - 1$ файлов.

Пример полного решения на Python 3:

```
n = int ( input () )
t = list ( map ( int , input () . split () ) )

ans = 0
sum_t = sum ( t )
downloaded = 0
for i in range (1 , n + 1) :
    downloaded += t [ i - 1]
    ans = max ( ans , abs ( i * 100 // n - downloaded * 100 // sum_t )
)

print ( ans )
```

В подзадаче 5 (все t_i одинаковы) процент файлов и процент времени всегда совпадают и, следовательно, ответ на задачу — 0.

Дабы не возникло проблем с точностью при округлении, стоит все операции выполнять в 64-битном целочисленном типе данных.

Задача D. Задача на делимость

Стандартный ввод. Стандартный вывод. 2 секунды. 256 мегабайт.

Вам задана строка s, состоящая из цифр и q запросов к этой строке.

Напомним, что подстрокой $s[l; r]$ строки s называется строка $s_l s_{l+1} \dots s_r$. Например, подстроками «0123456789» являются «345», «012», «4», «56789», но не строки «019» и «135».

Запросы имеют следующий вид:

- l r $(1 \le l \le r \le |s|)$: посчитать остаток от деления на 3 десятичного числа, образованного подстрокой $s[l; r]$.

Формат входных данных

Первая строка входных данных содержит одну строку s, состоящую из не более, чем $3*10^5$ цифр.

Вторая строка входных данных содержит одно целое число q $(1 \le q \le 3*10^5)$ – количество запросов.

Следующие q строк содержат запросы, по одному в строке. Каждый запрос задан в формате, описанном в условии задачи.

Формат выходных данных

На каждый запрос выведите ответ на него – остаток от деления на 3 десятичного числа, образованного цифрами из строки *s* лежащими в отрезке [*l; r*].

Система начисления баллов

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой

подзадачи и необходимых подзадач успешно пройдены.

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**	**Информация о проверке**
0	-	примеры из условия		полная
1	20	$\|s\|, q \le 10^3$		первая ошибка
2	30	Для всех запросов верно: *s*[*l; r*] содержит не более 1000 символов отличных от 0	1	первая ошибка
3	50	без дополнительных ограничений	1–2	первая ошибка

Пример

стандартный ввод	стандартный вывод
0123456789 5 8 8 1 10 3 7 4 4 2 6	1 0 2 0 0

Решение:

Докажем, что вне зависимости от того, на какой позиции стоит некоторая цифра *digit* она дает один и тот же остаток по модулю 3.

$(digit*10^k) \bmod 3 = ((digit*10^{k-1}) \bmod 3) * (10 \bmod 3)$
$10 \bmod 3 = 1$
$digit*10^k \bmod 3 = digit*10^{k-1} \bmod 3$

Из утверждения доказанного выше вытекает, что для ответа на запрос достаточно взять сумму всех цифр на отрезке по модулю 3.

Задача E. Атака на харвестер

Стандартный ввод. Стандартный вывод. 2 секунды. 512 мегабайт.

Представим пустыню Арракиса в виде клетчатого поля размера *n* x *m*, строки которого пронумерованы от *1* до *n* сверху вниз, а столбцы – от *1* до *m* слева направо.

Харконнены собирают в пустыне *пряность* – очень ценный ресурс, без которого невозможна, в том числе, космическая навигация. Изначально харвестер (огромный комбайн для сбора пряности) располагается в клетке *(1, 1)*, то есть в левой верхней клетке поля, после чего он может перемещаться в клетку, соседнюю по правой или нижней стороне с текущей. Иными словами, из клетки *(i, j)* харвестер может переместиться в *(i, j + 1)* или в *(i + 1, j)*, если эти координаты не указывают за границу поля.

Между некоторыми соседними по стороне клетками располагаются залежи пряности. Если харвестер перемещается между двумя такими клетками, за это перемещение он собирает ровно единицу пряности.

У маршрута харвестера есть определенная конечная клетка *(r, c)*. В этой клетке также сейчас располагается отряд фрименов под предводительством Пола Атрейдеса, готовящий нападение на харвестер. Далее события развиваются следующим образом:

1. Харконнены выбирают один из кратчайших маршрутов из *(1, 1)* в *(r, c)*, то есть любой из путей до конечной клетки длины $r + c - 2$.
2. Фримены узнают этот маршрут и готовятся выдвигаться по нему навстречу харвестеру.
3. Затем харвестер и отряд фрименов **по очереди** начинают совершать по одному перемещению в соседнюю клетку вдоль этого маршрута. Харвестер каждый раз движется на одну клетку вправо или вниз, а фримены, соответственно – влево или вверх. Первое перемещение совершает харвестер.
4. Процесс останавливается, когда харвестер и отряд фрименов оказываются в одной клетке.

Вам предстоит ответить на несколько запросов, i-й из которых задается конечной клеткой (r_i, c_i). Для каждого запроса выберите такой маршрут из *(1, 1)* в (r_i, c_i), при котором харвестер сможет собрать как можно больше пряности до того, как случится нападение фрименов.

Формат входных данных

В первой строке даны три целых числа n, m и k – размеры поля и количество залежей пряности на поле ($1 \le n, m, n * m \le 2*10^5$; $0 \le k \le 2nm - n - m$).

В i-й из следующих k строк даны четыре целых числа $A_{i,r}$, $A_{i,c}$, $B_{i,r}$ и $B_{i,c}$ – координаты клетки A_i и клетки B_i, на границе между которыми находятся i-е залежи пряности ($1 \le A_{i,r}, B_{i,r} \le n$; $1 \le A_{i,c}, B_{i,c} \le m$). Гарантируется, что клетки A_i и B_i являются соседними по стороне.

В следующей строке дано одно целое число q – количество запросов, на которые вам предстоит ответить ($1 \le q \le 2*10^5$).

В *i*-й из следующих *q* строк даны два целых числа r_i и c_i, описывающие *i*-й запрос – координаты конечной клетки маршрута харвестера, в которой изначально находится отряд фрименов ($1 \le r_i \le n$; $1 \le c_i \le m$).

Формат выходных данных

Выведите *q* строк, в *i*-й из которых выведите единственное целое число – максимальное количество единиц пряности, которое может добыть харвестер за половину пути до клетки (r_i, c_i).

Система начисления баллов

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**	**Информация о проверке**
0	-	пример из условия		полная
1	9	$n, m, q \le 10$	0	полная
2	11	$n = 1, m, q \le 1000$		первая ошибка
3	13	$n = 1$	2	первая ошибка
4	16	$n \le 2$	2, 3	первая ошибка
5	8	$k = 1$		первая ошибка
6	17	$q \le 20$	0	первая ошибка
7	26	нет	0 – 6	первая ошибка

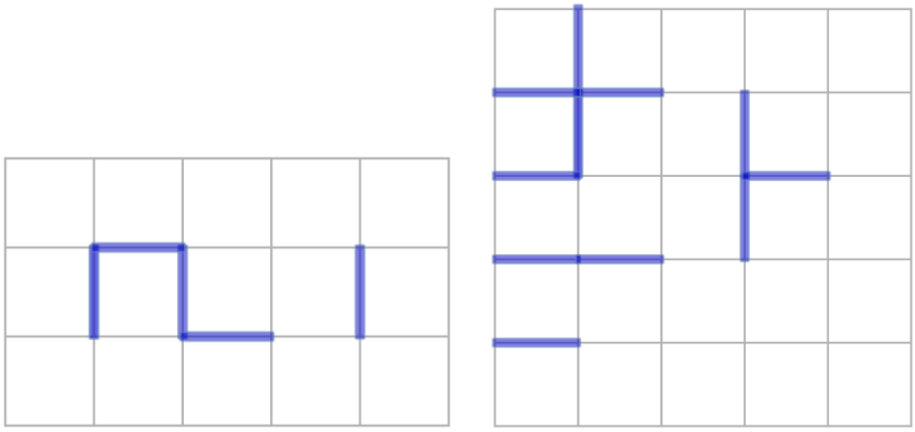
Примеры

стандартный ввод	стандартный вывод
3 5 5 2 1 2 2 1 2 2 2 2 2 2 3 2 3 3 3 2 4 2 5 5 1 1 2 3 1 4 3 3 3 5	0 1 0 1 2
5 5 11 2 2 2 1 2 2 1 2 2 1 3 1 3 1 4 1 5 1 4 1 3 2 4 2 3 3 3 4 3 4 2 4 2 4 2 3 1 1 1 2 1 1 2 1 5 5 5 2 3 2 1 4 1	4 2 1 2 1

1 5	

Замечания

Расположения залежей пряности для первого и второго примера изображены ниже:



Решение:

Подзадача 1

В первой подзадаче достаточно было проэмулировать процесс. Всего возможных путей до конечной клетки меньше, чем 2^{n+m-2}, что позволяло перебрать для каждого запроса все пути, проэмулировать перемещения и выбрать оптимальный.

Подзадачи 2–4

Во второй и третьей подзадачах дано, что $n = 1$, а значит таблица представляет собой полосу клеток. В такой полосе маршруты харвестера и фрименов всегда фиксированы, а также известна конечная клетка маршрута – ровно середина между $(1, 1)$ и $(1, c_i)$.

Тогда для ответа на запрос достаточно уметь находить количество залежей пряности на некотором префиксе этой полосы. Если считать их для каждого ответа заново, то можно пройти вторую подзадачу; если же предподсчитать префиксные суммы, решение проходило и третью.

В четвертой подзадаче решение не сильно отличается от описанного: в случае $r_i = 1$ решение идентично, а при $r_i = 2$ достаточно было заметить, что возможных мест встречи харвестера и отряда фрименов не более двух, а максимальное количество пряности, собранное по пути до каждого места, можно предподсчитать (например, описанным далее способом, но подошел бы и немного менее оптимальный).

Подзадача 5

Эта подзадача была рассчитана на частичное решение, проверяющее, мог ли харвестер собрать единственные залежи пряности на поле раньше, чем

встретился бы с отрядом фрименов. Обозначим клетку справа или снизу от залежей пряности за (r_s, c_s), тогда, если они могут быть собраны, должно выполняться

- $r_s \le r_i$ и $c_s \le c_i$, чтобы через эту клетку проходил хотя бы один кратчайший маршрут до (r_i, c_i);
- $(r_s - 1) + (c_s - 1) \le (r_i - r_s) + (c_i - c_s)$, чтобы харвестер мог добраться до этой клетки раньше, чем фримены.

Если оба условия выполняются, то ответ на задачу равен *1*, иначе – 0.

Полное решение

Для полного решения посчитаем для каждой клетки значение `dp`[i][j] - максимальное количество залежей пряности, которые могут быть встречены по пути из клетки *(1, 1)* в клетку *(i, j)*. Это значение можно посчитать с помощью динамического программирования как

`max(dp`[i][j–*1*] + `spice(L`, i, j), `dp`[i–*1*][j] + `spice(U`, i, j))

где `spice(dir`, i, j) равно *1*, если в направлении `dir` от клетки *(i, j)* находятся залежи пряности, и *0* иначе. Значения `spice` можно предподсчитать, считывая входные данные: для клетки *(i, j)* важны наличия во вводе четверки *(i – 1, j, i, j)* и *(i, j – 1, i, j)*.

Теперь для каждой клетки (r_i, c_i) заметим, что

- при движении навстречу харвестер произведет $d_i = \left\lceil \frac{r_i + c_i - 2}{2} \right\rceil$ перемещений, а фримены – $\left\lfloor \frac{r_i + c_i - 2}{2} \right\rfloor$ перемещений;
- на расстоянии d_i от клетки *(1, 1)* образуют отрезок некоторой диагонали таблицы;
- в зависимости от того, что больше – r_i или c_i, это будет либо отрезок диагонали, примыкающий к верхней стороне таблицы, либо отрезок диагонали, примыкающий к левой стороне таблицы.

Для решения шестой подзадачи можно было просто перебрать все эти клетки, которые находятся на середине пути до (r_i, c_i) и выбрать максимум из значений `dp` в них.

А для полного решения можно воспользоваться описанными свойствами этих клеток и предподсчитать максимум значений `dp` на каждом «префиксе» и каждом «суффиксе» каждой диагонали. Предподсчитанные значения позволяют получать ответ на каждый запрос за *O(1)* времени.