Первый отборочный этап

Первый отборочный тур проводится индивидуально в сети Интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Для каждой из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по математике, задачи по информатике общие для всех участников. Решение задач по информатике предполагало написание программ. Участники не были ограничены в выборе языка программирования для решения задач. На решение задач каждого предмета первого отборочного этапа участникам давалось 2 дня. У участников было два временных слота по 2 дня каждый, когда они могли решать задачи по предмету. Решение каждой задачи дает определенное количество баллов.

Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (математика и информатика) — суммарно от 0 до 200 баллов.

Задачи первого этапа. Информатика

Первая попытка

Задача I.1.1.1. Фанатам стратегий (20 баллов)

Миссия простая: нужно либо накопить c кредитов (читайте: единиц вымышленной валюты), либо сокрушить врага в этом регионе. Миссия проходная и не интересная (даже никаких юнитов кроме пехоты), так что надо её закрыть побыстрее.

Изначально у нас нет ни одного кредита и ни одного отряда пехоты. Но есть база и гарантия того, что враг нас не обнаружит, пока мы на него не нападём.

На нашей базе можно хранить не более x кредитов. Чтобы хранить больше, нужно строить хранилища: каждое стоит y ($y \leqslant x$) кредитов и в каждом можно будет хранить дополнительно по максимум z кредитов. Строительство моментально (а вы что, реализма ожидали?).

Через каждые m минут нам привозят добычу на s кредитов. Добыча моментально конвертируется в кредиты и мы перераспределяем новые s кредитов и кредиты, имеющиеся на базе, по трём направлениям: строительство хранилищ, наём отрядов пехоты, хранение на базе. Количество кредитов, отданных на строительство хранилищ, должно быть кратно стоимости постройки одного хранилища. Количество кредитов, отданных на наём отрядов пехоты, должно быть кратно стоимости наёма одного отряда пехоты. Если мы отдаём на хранение больше кредитов, чем может храниться на базе (хранилища, на строительство которых мы только что отдали кредиты, тоже учитываются), то излишки кредитов исчезают.

Если после очередного описанного выше распределения на базе будет храниться хотя бы c кредитов, то считается, что мы накопили c кредитов, и миссия считается пройденной.

Один отряд пехоты стоит f ($f \leqslant x$) кредитов. Сразу после оплаты, которая тоже

моментальна, отряд будет ждать приказаний на нашей базе.

Проще и быстрее всего избавиться от врага в этом регионе – заставить его капитулировать. Для этого нужно привести к стенам вражеской базы больше отрядов пехоты, чем есть на вражеской базе.

По данным разведки, на вражеской базе e отрядов, а добраться до вражеской базы наши отряды пехоты смогут за t минут (все отряды, независимо от их количества, могут идти вместе).

За сколько в лучшем случае мы завершим миссию?

Формат входных данных

Единственная строка содержит 9 целых чисел $c, x, y, z, m, s, f, e, t$ ($1 \leqslant c, x, y, z, m, s, f, e, t \leqslant 10^9$; $f, y \leqslant x$).

Формат выходных данных

Выведите одно целое число – искомое количество минут.

Пояснения

Чтобы максимально быстро накопить 2700 кредитов, мы можем построить 2 хранилища через 3 минуты после начала (в этот момент у нас появятся первые кредиты) и далее просто складировать все кредиты на базе. Тогда через 9 минут после начала на базе накопится необходимая сумма.

Чтобы заставить врага сдаться, мы можем через 3 минуты после начала нанять 10 отрядов пехоты и сразу же пойти на вражескую базу. Тогда через 6 минут после начала враги сдадутся.

Примеры

Пример №1

Стандартный ввод
`2700 1000 150 1000 3 1000 100 7 3`
Стандартный вывод
`6`

Решение

В данной задаче нужно выбрать быстрейший путь: накопить c кредитов или сокрушить врага.

Рассмотрим накопление. В этом случае для минимизации времени стоит тратить кредиты только на постройку хранилищ. Так как стоимость одного хранилища не превышает изначального максимального количества кредитов, которое мы можем хранить на базе, то есть $y \leqslant x$, при поставке добычи мы всегда можем распределить кредиты так, что мы отдаём на хранение меньше кредитов, чем может храниться на базе.

Таким образом, если нужны хранилища, то есть $c > x$, достаточно построить $\left\lceil \frac{c-x}{z} \right\rceil$ хранилищ. Получается, нужно набрать в сумме $c + \left\lceil \frac{\max(0, c-x)}{z} \right\rceil \cdot y$ кредитов, что мы сделаем за $\left\lceil \frac{c+\left\lceil \frac{\max(0, c-x)}{z} \right\rceil \cdot y}{s} \right\rceil \cdot m$ минут.

Рассмотрим капитуляцию врага. В этом случае для минимизации времени стоит тратить кредиты только на наём отрядов пехоты. Так как стоимость одного отряда пехоты не превышает изначального максимального количества кредитов, которое мы можем хранить на базе, то есть $f \leqslant x$, при поставке добычи мы всегда можем распределить кредиты так, что мы отдаём на хранение меньше кредитов, чем может храниться на базе.

Чтобы заставить врага капитулировать, достаточно нанять $e + 1$ отрядов пехоты и привести все эти отряды к стенам вражеской базы. Для этого придётся потратить в сумме $(e + 1) \cdot f$ кредитов на пехоту, что мы сделаем за $\left\lceil \frac{(e+1) \cdot f}{s} \right\rceil \cdot m$ минут. Плюс t минут на то, чтобы добраться до врага. Итого $\left\lceil \frac{(e+1) \cdot f}{s} \right\rceil \cdot m + t$ минут.

Минимум из двух описанных значений и является ответом. Также стоит отметить, что 64-битных типов данных для полного решения будет недостаточно.

Пример программы-решения

Ниже представлено решение на языке Python3

```
c, x, y, z, m, s, f, e, t = [int(i) for i in input().split()]
print(min((c + (max(0, c - x) + z - 1) // z * y + s - 1) // s * m,
          ((e + 1) * f + s - 1) // s * m + t))
```

Задача I.1.1.2. Настроение профессора (20 баллов)

Все персонажи и описываемые события являются вымышленными. Любое совпадение с реальными людьми или событиями случайно.

Завтра студенты первого курса пойдут сдавать математический анализ. Экзамен будет принимать профессор Ильдар.

Экзамен будет проходить по старинке: студенты по одному подходят к профессору, отвечают на заданные им вопросы и получают свои оценки. Результат экзамена сильно зависит от настроения профессора Ильдара: если у него плохое настроение, то не важно, насколько хорошо вы подготовились, – он отправит вас на пересдачу.

Пусть настроение профессора в некоторый момент времени равно x. После ответов отличника настроение профессора повышается и становится равно $x + 1$. После ответов хорошиста настроение профессора не меняется. А если ответы явно не тянут на оценку 4, то профессор ставит 3 и его настроение падает до $x - 1$.

Но если завтра в какой-либо момент времени настроение профессора будет равно отрицательному числу, то после этого момента описанные выше закономерности перестают действовать и все студенты, что ещё не получили своих оценок, отправляются на пересдачу.

Сегодня вы (неожиданно) – староста группы и хотите, чтобы никто из ваших студентов не отправился на пересдачу. Порядок, в котором студенты будут подходить

к профессору, уже сформирован и его изменить нельзя, но вы знаете, насколько хорошо подготовился каждый из студентов, и знаете про профессора Ильдара ещё одну вещь – он любит шоколад.

Вы можете купить шоколадку (а лучше не одну) и подарить её профессору сегодня вечером. Каждая подаренная профессору шоколадка повышает его настроение на 1. Что профессор делает с шоколадками, никому не известно.

Какое минимальное количество шоколадок вам надо сегодня подарить профессору, чтобы завтра все студенты сдали экзамен?

Формат входных данных

В первой строке вводятся два целых числа n и k ($1 \leqslant n \leqslant 2 \cdot 10^5$, $-10^9 \leqslant k \leqslant 10^9$) – количество студентов в вашей группе и настроение профессора сегодня вечером (настроение профессора до начала экзамена может измениться только благодаря вам).

Во второй строке вводится строка из n символов a_i ($a_i \in \{A, B, C\}$). Эта строка описывает порядок, в котором студенты будут подходить к профессору. Каждый студент описывается одним символом. Символом A обозначается отличник, символом B – хорошист, символом C – троечник или неподготовившийся к экзамену студент.

Формат выходных данных

Выведите одно целое число – искомое минимальное количество шоколадок.

Примеры

Пример №1

Стандартный ввод
3 0 BCA
Стандартный вывод
1

Пример №2

Стандартный ввод
3 3 AAA
Стандартный вывод
0

Решение

Промоделируем сдачу экзамена без учёта отправки всех на пересдачу. Найдём минимальное настроение профессора в день экзамена (учитывая настроение до начала и не учитывая после окончания). Если оно отрицательно и равно x, то достаточно купить профессору $-x$ шоколадок перед экзаменом, чтобы никто не был отправлен на пересдачу. Если оно неотрицательно, то можно обойтись без подарков профессору.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, k = [int(i) for i in input().split()]
min_k = 10 ** 9 + 1
for c in input():
    min_k = min(k, min_k)
    k += ord('B') - ord(c)

print(max(0, -min_k))
```

Задача I.1.1.3. Комната ярости (20 баллов)

Гертруда имеет n тарелок. И хочет разбить их все. По одной. Но тарелки бьются очень звонко. Она опасается, что повредит слух.

Известно, что сила звона первой разбитой тарелки будет равна a_1. Сила звона каждой последующей разбитой тарелки будет в b раз больше силы звона предыдущей. То есть сила звона i-ой $(i > 1)$ разбитой тарелки будет равна $a_i = a_{i-1} \cdot b$.

Гертруда знает максимальное суммарное значение сил звона MAX, которое могут выдержать её уши, и желает максимально насладиться звуками бьющейся посуды.

Помогите Гертруде, найдите максимальное количество тарелок, которые она может разбить, не повредив слух. И побыстрее.

Формат входных данных

В первой строке вводятся четыре целых числа n, a_1, b, MAX $(1 \leqslant n, MAX \leqslant 10^{1000}, 1 \leqslant a_1, b \leqslant 10)$.

Формат выходных данных

Выведите максимальное количество тарелок.

Пояснения

Если Гертруда разобьёт одну тарелку, то суммарное значение сил звона будет равно $a_1 = 1$.

Если разобьёт две тарелки, то суммарное значение будет равно $a_1 + a_2 = 1 + 2 = 3$.

Если разобьёт три, то – $a_1 + a_2 + a_3 = 1 + 2 + 4 = 7$.

Примеры

Пример №1

Стандартный ввод
10 1 2 4

Стандартный вывод
2

Решение

В данной задаче для нахождения ответа можно было воспользоваться формулой суммы первых k элементов геометрической прогрессии и бинпоиском по ответу, но ограничения на входные данные устроены так, что можно было просто промоделировать разбивание тарелок по одной.

Для работы с длинными целыми числами стоило либо использовать соответствующий встроенный тип данных, либо писать свою реализацию длинной арифметики.

Первые подзадачи решались без длинной арифметики.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, ai, b, max = [int(i) for i in input().split()]

if b == 1:
    print(min(max // ai, n))
    exit(0)

# b > 1
l, s = 1, ai
while l <= n:
    if s > max:
        print(l - 1)
        exit(0)
    ai *= b
    s += ai
    l += 1

print(n)
```

Задача I.1.1.4. Подлизы (20 баллов)

Жила-была девочка Катя, и было у неё много-много денег. И подруг. Ну как подруг...

И собрались они как-то раз все вместе у Кати дома и обсуждали фильмы. Многие хвалили вкус Кати. Редко кто не соглашался с её мнением. О вкусах, конечно, не спорят, но Кате показалось это странным и она решила устроить проверку.

Катя записала m пар фильмов, которые девочки сравнивали, и для каждой такой пары пометила, какой из фильмов девочки посчитали однозначно лучше другого. А потом воспользовалась своим обаянием влиянием и убедила школьного психолога провести тестирование, в котором есть вопрос о трёх любимых фильмах. Вот так вот всё просто, когда ты – Катя.

Среди неиспорченных бланков тестирования (не спрашивайте, как она их достала) Катя нашла заполненные бланки n своих подруг. Скажите, согласовываются ли записи Кати с каждым из ответов на вопрос о трёх любимых фильмах в отдельности.

Формат входных данных

В первой строке заданы числа n и m ($1 \leqslant n, m \leqslant 10^5$).

В следующих m строках – пары фильмов, записанные у Кати. Первый фильм в паре считается лучше второго.

В следующих n строках – списки любимых фильмов девочек. Первый фильм в тройке считается лучше второго, а второй – лучше третьего.

Записи Кати непротиворечивы. Каждая пара фильмов в записях Кати встречается не более одного раза.

Так сложилось, что все фильмы, что встречаются в списках любимых фильмов девочек, встречаются и в записях Кати, а в каждом отдельно взятом списке все три фильма различны.

Для вашего же удобства названия фильмов во входных данных заменены на положительные натуральные числа, не превышающие 10^6.

Формат выходных данных

Выведите n строк, в i-ой из которых должно быть написано *honest*, если список любимых фильмов из i-го бланка не противоречит записям Кати, или *liar*, если противоречит.

Не выводите лишние пробелы в конце или начале строк - это будет считаться за ошибку.

Пояснения

Тройка фильмов 1 2 4 противоречит записям Кати, так как по записям Кати фильм 5 лучше фильма 4, но его нет в тройке.

Тройка фильмов 1 3 2 противоречит, так как по записям Кати фильм 2 лучше фильма 3, а в тройке фильм 3 стоит до фильма 2.

Тройка фильмов 5 4 8 противоречит, так как по записям Кати фильм 2 лучше фильма 4, но его нет в тройке.

Примеры

Пример №1

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

Стандартный вывод
honest liar liar liar honest

Решение

Из условия следует, что тройка любимых фильмов подруги Кати может противоречить записям Кати одним из двух способов (или сразу обоими способами):

1. Существует фильм, который лучше одного из тройки, но в эту тройку не включён;
2. 2 фильма из тройки сравнивали и фильм, который хуже по записям Кати, оказался в тройке на месте выше, чем фильм, который лучше по записям Кати.

Также можно вывести определение того, какая тройка фильмов не противоречит записям Кати.

Такая тройка удовлетворяем всем следующим условиям:

1. Не существует фильма, который по записям Кати лучше первого фильма в тройке;
2. Фильмов, которые по записям Кати лучше второго фильма в тройке,
   либо не существует,
   либо такой фильм только один и это первый фильм в тройке;
3. Фильмов, которые по записям Кати лучше третьего фильма в тройке,
   либо не существует,
   либо такой фильм только один и это первый фильм в тройке,
   либо такой фильм только один и это второй фильм в тройке,
   либо таких фильм ровно два и это первый и второй фильмы в тройке.

Остаётся лишь аккуратно реализовать проверку каждой из троек фильмов.

Асимптотика: $O(m+n)$

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, m = [int(i) for i in input().split()]

better = [[] for i in range(int(1e6) + 1)]

for i in range(m):
    film1, film2 = [int(i) for i in input().split()]
    better[film2].append(film1)

for i in range(n):
    film1, film2, film3 = [int(i) for i in input().split()]
    ok = len(better[film1]) == 0
    for f in better[film2]:
        if not f == film1:
            ok = False
            break
    for f in better[film3]:
        if (not f == film1) and (not f == film2):
            ok = False
            break
    print("honest" if ok else "liar")
```

Задача I.1.1.5. Циклические сдвиги vs разворот строки (20 баллов)

На этой неделе на уроках информатики Васе рассказывают про строки.

Вчера Вася узнал, что такое циклический сдвиг:

k-й циклический сдвиг строки – это строка, полученная перестановкой первых k символов строки в её конец. В частности, 0-й циклический сдвиг строки – это сама строка.

И он написал программу, которая умеет перемещать первый символ строки в её конец k раз, получая таким образом k-й циклический сдвиг строки.

Сегодня Васе нужно реализовать разворот строки. Но у Васи тренировка. Ему некогда писать новые сложные программы. Поэтому он задался вопросом:

Можно ли циклическими сдвигами развернуть строку?

Благодаря своим друзьям Вася узнал, на какой строке (да, всего одной) будет тестировать его программу учитель, у которого нет времени рецензировать код каждого ученика. Поэтому вопрос упростился:

Можно ли циклическими сдвигами развернуть строку s?

Помогите ему ответить на этот вопрос.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 3 \cdot 10^5$) – длина строки s.

Во второй строке – сама строка s, на которой будет тестировать программу Васи учитель. Состоит строка s только из строчных латинских букв.

Формат выходных данных

Если строку нельзя развернуть циклическими сдвигами, то выведите число -1. В противном случае выведите такое целое число k ($0 \leqslant k < n$), что k-й циклический сдвиг строки s равен развёрнутой строке s. Если таких k несколько, выведите любое из них.

Пояснения

0-й циклический сдвиг строки s равен $abac$.

1-й циклический сдвиг строки s равен $baca$.

2-й циклический сдвиг строки s равен $acab$.

3-й циклический сдвиг строки s равен $caba$.

Развёрнутая строка s равна $caba$.

Единственное подходящее k равно трём.

Примеры

Пример №1

Стандартный ввод
4 abac
Стандартный вывод
3

Решение

Условие задачи можно свести к одному вопросу: можно ли циклическими сдвигами развернуть строку s?

Для нахождения ответа можно было искать развёрнутую строку s в строке $t = s + s$, так как все различные циклические сдвиги строки s являются подстроками строки t.

Наивная реализация этого поиска имеет асимптотику $O(n^2)$ по времени. За $O(n)$ по времени данную задачу можно было решить, например, с помощью алгоритма Кнута-Морриса-Пратта.

Другой способ решения: проверим, что строку s можно разрезать на два палиндрома. Быстро сделать это можно было, например, с помощью алгоритма Манакера.

Пример программы-решения

Ниже представлено решение на языке Python3

```python
def pi(s):
    n = len(s)
    pi = [0] * n
    j = 0
    for i in range(1, n):
        while j > 0 and s[i] != s[j]:
            j = pi[j - 1]
        if s[i] == s[j]:
            j += 1
        pi[i] = j
    return pi


n = int(input())
s = input()

pi = pi(s[::-1] + '#' + s + s)
for i in range(n):
    if pi[n * 2 + i] == n:
        print(i)
        break
else:
    print(-1)
```

Вторая попытка

Задача I.1.2.1. Фанатам стратегий 2 (16 баллов)

В текущей миссии, очевидно, необходимо укрепить базу, прежде чем идти в открытый бой.

Для обеспечения устойчивой обороны требуется построить n различных новых зданий. Но не всё так просто.

Для поддержания процессов, которые будут происходить в этих зданиях, необходимо электричество. А получать электроэнергию новые здания могут только от новых электростанций. Новых электростанций на базе нет, так что их тоже придётся построить.

Зная, сколько единиц электроэнергии в единицу времени производит одна новая электростанция и количество электроэнергии, потребляемое за единицу времени каждым из упомянутых выше n новых зданий, определите минимальное количество электростанций, которое необходимо для полного функционирования требуемых n зданий.

Примечание: Считается, что электростанции не потребляют электроэнергии и среди n различных новых зданий, которые требуется построить, нет электростанции.

Формат входных данных

В первой строке заданы числа n и e ($1 \leqslant n \leqslant 10^5, 1 \leqslant e \leqslant 10^9$) – количество требуемых зданий и количество единиц электроэнергии, которое производит одна новая электростанция.

Во второй строке даны n чисел – количество единиц электроэнергии, потребля-

емое за единицу времени каждым из зданий. Все числа во второй строке неотрицательны и не превышают 10^9.

Формат выходных данных

Выведите одно целое число – минимальное количество электростанций, которое необходимо для полного функционирования требуемых n зданий.

Пояснения к примеру

Двух электростанции явно мало. Электроэнергию трёх электростанций можно распределить по зданиям следующим образом: на первое здание идёт 5 единиц от первой электростанции, на второе – 8 единиц от второй электростанции и 4 единицы от третьей, а на третье – 3 единицы от первой электростанции и 4 единицы от третьей.

Примеры

Пример №1

Стандартный ввод
3 8 5 12 7
Стандартный вывод
3

Решение

Пусть sum – суммарное количество единиц электроэнергии, потребляемое за единицу времени описанными n зданиями. Тогда, чтобы эти здания работали, нужно построить $\left\lceil \frac{sum}{s} \right\rceil$ новых электростанций.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, s = [int(i) for i in input().split()]
sm = sum(int(i) for i in input().split())
print((sm + s - 1) // s)
```

Задача I.1.2.2. I Don't Like (20 баллов)

Кому-то не нравятся наши задачи. Наверно, из-за их сложности. Кто-то ругает нас за то, что программа не компилируется на компиляторах, имеющихся на Stepik, или просто не проходит наши тесты к задаче, хотя у кого-то на компьютере всё работает. Кто-то, не указывая на недочёты в задаче, хочет, чтобы ему или ей разжевали условие задачи, и после возмущается, прочитав, что мы не делаем пояснений и кратких пересказов условий, так как некорректностей найдено не было и мы хотим оставить всех участников олимпиады в равных условиях. А кто-то считает, что его

тесты к задаче не хуже тех, что создали мы, и его решение верно, так как на его тестах оно работает (да, и такие есть). А кто-то списывает.

Всем этим замечательным людям мы можем лишь пожелать здоровья и бесконечного количества нервных клеток. Смириться с правилами олимпиады тоже не помешает.

А маленькому Коле не нравится, когда числа в массиве не отсортированы по возрастанию (если быть точным, по неубыванию, но Коля таких слов не знает).

Вот кто придумал дарить детям неотсортированные массивы? Мы не знаем, но Коля сегодня получил именно такой подарок. Он даже решил посчитать число таких пар индексов массива (i, j), что $i < j$ и $a_i > a_j$, чтобы хоть как-то измерить силу своей ненависти к подаренному ему массиву a и тому человеку, который это сделал.

Коля устал злиться, но сумеет сделать ещё ровно одно действие – поменять два элемента массива a местами. Ручки у него короткие, так что Коля может поменять местами только соседние элементы массива a (то есть такие элементы, индексы которых различаются не более чем на 1).

Определите количество способов, которыми Коля может уменьшить описанное выше число пар индексов. Два способа считаются различными, если существует индекс, который встречается только в одной из двух пар индексов, описывающих эти два способа.

Формат входных данных

В первой строке задано число n ($1 \leqslant n \leqslant 10^5$) – количество элементов в массиве a.

Во второй строке даны n чисел a_i ($-10^9 \leqslant a_i \leqslant 10^9$) – элементы массива a.

Гарантируется, что числа в массиве a не упорядочены по неубыванию.

Формат выходных данных

Выведите одно целое число – количество способов, которыми Коля может уменьшить описанное выше число пар индексов.

Примеры

Пример №1

Стандартный ввод
3 1 3 2

Стандартный вывод
1

Решение

Как изменится описанное в условии число пар индексов, если мы поменяем местами элементы a_i и a_{i+1} ($1 \leqslant i < n$)? Если $a_i > a_{i+1}$, то оно уменьшится на единицу; если $a_i < a_{i+1}$, то – увеличится на единицу; если $a_i = a_{i+1}$, то – не изменится.

Таким образом, в данной задаче достаточно посчитать количество пар соседних элементов в массиве a, в которых $a_i > a_{i+1}$.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n = int(input())
a = [int(i) for i in input().split()]
print(sum(1 if a[i] > a[i + 1] else 0 for i in range(n - 1)))
```

Задача I.1.2.3. Мультимножества (20 баллов)

Дан набор из n чисел. Каждое число отнесли ровно к одному из 5-и мультимножеств: A, B, C, D или E.

По итогу такого распределения чисел получилось так, что все 5 мультимножеств непусты, суммы элементов мультимножеств равны и соблюдается следующее условие:

Для любых $a \in A$, $b \in B$, $c \in C$, $d \in D$ и $e \in E$ выполняется неравенство $a \leqslant b \leqslant c \leqslant d \leqslant e$.

Определите, правда ли, что такое могло произойти.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 10^5$) – размер набора чисел.

Вторая строка содержит n целых чисел a_i ($-10^9 \leqslant a_i \leqslant 10^9$) – сами числа набора.

Формат выходных данных

Выведите Yes, если возможно разбиение данных n чисел на мультимножества. Иначе выведите No.

Примеры

Пример №1

Стандартный ввод
19 2 1 1 2 2 0 2 3 11 3 3 4 3 4 0 6 5 1 2

Стандартный вывод
Yes

Решение

Пусть sum – сумма всех чисел данного набора, а $\max(X)$ и $\min(X)$ – наибольший и наименьший элементы множества X соответственно.

Тогда сумма элементов каждого из мультимножеств в отдельности равна $sum/5$ и описанное условие эквивалентно следующему:

$$\max(A) \leqslant \min(B) \leqslant \max(B) \leqslant \min(C) \leqslant \max(C) \leqslant \min(D) \leqslant \max(D) \leqslant \min(E)$$

Следовательно, для нахождения возможных мультимножеств стоит жадно добавлять наименьшие числа из набора в мультимножество A, пока сумма его элементов не достигнет $sum/5$, потом в мультимножество B, пока сумма его элементов не достигнет $sum/5$, и так далее. Если получилось сформировать мультимножества, удовлетворяющие всем условиям, то ответ `Yes`, иначе – `No`.

Асимптотика: $O(n \cdot \log(n))$

Пример программы-решения

Ниже представлено решение на языке Python3

```python
n = int(input())
a = [int(i) for i in input().split()]
a.sort()
sum_a = sum(a)

if sum_a % 5 != 0:
    print('No')
    exit(0)

cur, cnt = 0, 1
for i in a:
    cur += i
    if cur == sum_a // 5 * cnt:
        cnt += 1
        if cnt == 6:
            break

print('Yes' if cnt == 6 else 'No')
```

Задача I.1.2.4. Фанатам стратегий 3 (24 баллов)

Данная задача – логическое продолжение задачи "Фанатам стратегий 2". Рекомендуем перед решением данной задачи полностью решить задачу "Фанатам стратегий 2".

Вскоре стало понятно, что всё совсем не просто. Нельзя взять и построить здание. Их в этой игре ещё и открыть нужно.

Новое здание типа A можно построить, только если на нашей базе функционирует хотя бы по одному новому зданию из списка необходимых зданий здания типа A.

Сколько на самом деле нам придётся построить зданий (не считая электростанций)? Какие они? В каком порядке их строить? Ваша задача – найти ответы на эти вопросы.

Примеры

Гарантируется, что существует такая последовательность постройки зданий, что здания всех типов можно построить.

Формат входных данных

В первой строке записаны три целых числа n, m и t ($1 \leqslant m \leqslant n \leqslant 5 \cdot 10^4$; $1 \leqslant t \leqslant 2$) – количество различных типов новых зданий в игре, количество новых зданий, которые нужно построить, и номер формата выходных данных.

В следующей строке записаны m названий типов зданий, разделённых пробелами – требуемые для обеспечения устойчивой обороны здания. Гарантируется, что строка не содержит одинаковых типов зданий.

Далее идёт n блоков по 2 строки следующего вида:

В первой строке – название типа здания.

Во второй – длина списка необходимых зданий для здания данного типа и сам список необходимых зданий. Гарантируется, что список не содержит одинаковых типов зданий.

Сумма длин списков необходимых зданий не превышает $5 \cdot 10^4$.

Название каждого типа здания состоит только из латинских букв и имеет длину не более десяти символов.

Формат выходных данных

Если $t = 1$, то выведите одно число – минимальное количество зданий, которые нужно построить.

Если $t = 2$, то в первой строке выведите одно число – минимальное количество зданий, которое необходимо построить, а во второй – k названий зданий, которые нужно построить, в том порядке, в котором их нужно строить. Если существует несколько подходящих последовательностей – выведите любую из них.

Примеры

Пример №1

Стандартный ввод
13 5 2 refinery vehicle repair palace turret constryard 0 windtrap 1 constryard refinery 1 windtrap outpost 1 windtrap silo 2 refinery constryard vehicle 3 refinery windtrap outpost barracks 2 constryard outpost wall 1 outpost turret 1 outpost starport 2 silo refinery repair 1 vehicle hitech 3 vehicle wall outpost palace 1 starport

Стандартный вывод
10 constryard windtrap refinery outpost silo vehicle turret starport repair palace

Решение

Если представить, что здание типа A – это вершина орграфа, а список необходимых зданий здания типа A – список рёбер, направленных в вершину-здание типа A из других вершин-зданий, то задачу можно свести к задаче о нахождении топологической сортировки, с той лишь разницей, что вывести требуется не все вершины орграфа, а только те, из которых доступна хотя бы одна из m выделенных вершин-зданий.

Вершины-здания, которые нужно выводить, можно определить, например, с помощью серии обходов в глубину из m выделенных вершин-зданий, предварительно развернув в орграфе все рёбра. Порядок, в котором можно выводить эти вершины, может задать топологическая сортировка этих вершин. Топологическую сортировку можно найти, например, с помощь алгоритма Кана или алгоритма Тарьяна.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, m, tp = [int(i) for i in input().split()]
need = set(input().split())
lists = {}
rev_lists = {}

for i in range(n):
    s = input()
    lists.setdefault(s, [])
    rev_lists.setdefault(s, [])
    for t in input().split()[1:]:
        lists.setdefault(t, [])
        lists[t].append(s)
        rev_lists[s].append(t)

q = [""] * n
ql = qr = 0
for s in lists.keys():
    if len(rev_lists[s]) == 0:
        q[qr] = s
        qr += 1

cnt = {}
while ql < qr:
    s = q[ql]
    ql += 1
    for t in lists[s]:
        cnt.setdefault(t, 0)
        cnt[t] += 1
        if cnt[t] == len(rev_lists[t]):
            q[qr] = t
            qr += 1

for s in reversed(q):
    if s in need:
        for t in rev_lists[s]:
            need.add(t)

ans = []
for s in q:
    if s in need:
        ans.append(s)

print(len(ans))
if tp == 2:
    print(' '.join(map(str, ans)))
```

Задача I.1.2.5. Тир (20 баллов)

Давно были в тире? Мы недавно.

В нашем тире висят и стоят жестяные и алюминиевые банки из под различных напитков. Точнее, висели и стояли.

От наших выстрелов банки мотались из стороны в сторону на верёвке, срывались, звенели, мялись. Это вам не из пальцев стрелять.

Каждая из пуль либо прошла насквозь одной из банок, после чего поражённая банка упала на пол и откатилась в сторону так, что в неё было уже невозможно попасть; либо не попала ни в одну из банок. В любом случае, каждая из пуль застряла в стене, стоящей позади наших банок-мишеней.

Но тот день в прошлом. Осталась только стена с застрявшими в ней пулями и фотография. В попытке восстановить тот день и насладиться им снова мы собрали данные о положении каждой пули в стене, расположении банок и порядке выстрелов.

Помогите определить про каждую пулю, поразила ли она какую-то из банок, и если поразила, то какую именно.

Формат входных данных

В первой строке записаны два целых числа n и m $(1 \leqslant m,\ n \leqslant 10^5)$ – количество банок, которые были нашей мишенью в тот день, и количество совершённых в тот день выстрелов.

В i-ой из следующих n строк описывается положение i-ой банки. Положение задаётся координатами проекции банки на вертикальную плоскость. Проекция представляет из себя прямоугольник, стороны которого параллельны нанесённой на эту плоскость системы координат. Ось Y этой системы направлена вертикально вверх, а ось X – горизонтально. А прямоугольник задаётся парой точек – своей левой нижней и правой верхней вершинами.

Гарантируется, что ни одна пара этих прямоугольников не имеет ни одной общей точки.

В i-ой из следующих m строк описывается положение i-ой пули в стене. Пули заданы в том же порядке, в котором они выходили из наших дул. Сама стена строго вертикальна, поэтому мы можем считать, что положение задаётся координатами проекции пуль на вертикальную плоскость. Причём траектории движения пуль были строго перпендикулярна этой плоскости. Сами точки задаются парой координат в уже описанной выше системе координат.

Расстояние между банками и стеной по сравнению с расстоянием до стреляющих настолько мало, что мы им пренебрегаем.

Примечание: Значения координат по модулю не превышают 10^9.

Формат выходных данных

В первой и единственной строке выведите m чисел, i-ое из которых говорит, какую из банок i-я пуля прошла насквозь, если такая имеется. Если i не задела ни одну банку, то выведите -1, иначе выведите порядковый номер во входных данных банки, которую поразила i-я пуля.

Примеры

Пример №1

Стандартный ввод
4 10 0 0 1 1 2 3 3 8 15 15 20 20 10 12 12 13 2 2 0 -1 23 18 13 12 10 13 16 16 17 17 3 5 3 5 3 3

Стандартный вывод
-1 -1 -1 -1 4 3 -1 2 -1 -1

Решение

Один из способов решения:

Разобьём каждый прямоугольник на вертикальные отрезки: открывающий (с меньшей x координатой) и закрывающий (с большей x координатой).

Получившиеся отрезки отсортируем и будем обходить по неубыванию x координаты («запустим вертикальную сканирующую прямую от $-\infty$ до $+\infty$»). При равенстве x координаты сначала будем обрабатывать открывающие отрезки.

При обработке открывающего отрезка мы добавляем в некоторую структуру информацию о том, что сканирующая прямая на данный момент пересекает соответствующий отрезку прямоугольник. При обработке закрывающего отрезка мы удаляем из той же структуры информацию о том, что сканирующая прямая на данный момент пересекает соответствующий отрезку прямоугольник.

Так как прямоугольники не имеют общих точек, при любом положении сканирующей прямой открывающие отрезки, которые ещё «не закрыли», будут образовывать множество непересекающихся отрезков.

Следовательно можно множество открытых отрезков поддерживать, например, в set'е.

Если отсортировать точки-пули по неубыванию x координаты и параллельно обходить и их тоже, то мы сможем для каждой точки однозначно определить соответствующий набор открывающих отрезков; остаётся лишь найти ближайший к этой точке по y координате отрезок и проверить, лежит ли по y координате эта точка на нём. Для быстрого определения этого в set'е имеет смысл хранить открывающие отрезки в порядке, например, возрастания их «нижних» y координат и пользоваться функцией `lower_bound`.

Однако, так как мы изменили изначальный порядок точек-пуль, в случае если мы нашли для точки-пули прямоугольник, в который она попадает, это не значит, что данная пуля поразила соответствующую этому прямоугольнику банку. В проекцию банки могли попасть несколько проекций пуль, но только пуля, которая была выпущена раньше других, поразила банку.

Давайте для каждой проекции банки сохраним список проекций пуль, которые в него попадают. Тогда после окончания движения сканирующей прямой мы сможем определить для каждой банки, поразила ли её какая-либо пуля, и если поразила, то какая именно. А по этим данным мы можем восстановить ответ на задачу.

Асимптотика: $O(n \cdot \log(n))$

Пример программы-решения

Ниже представлено решение на языке C++

```cpp
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

#define P pair
#define fi first
#define se second

#define V vector
typedef V<int> Vi;
typedef V<ll> Vll;

#define all(v) (v).begin(), (v).end()

#define forn(i, n) for (int (i) = 0; (i) < (n); (i)++)

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);

    int n, m;
    cin >> n >> m;
    Vll leftX(n), downY(n), rightX(n), upY(n);
    forn(i, n) {
        cin >> leftX[i] >> downY[i] >> rightX[i] >> upY[i];
    }
    Vll x(m), y(m);
    forn(i, m) {
        cin >> x[i] >> y[i];
    }

    V<P<ll, int>> s(n * 2 + m);
    forn(i, n) {
        s[i * 2] = {leftX[i], -i - 1};
        s[i * 2 + 1] = {rightX[i], m + i};
    }
    forn(i, m) {
        s[n * 2 + i] = {x[i], i};
```

```
    }
    sort(all(s));

    Vi killed_by(n);
    fill(all(killed_by), INT_MAX);

    map<ll, int> mp;

    forn(i, n * 2 + m) {
        if (s[i].se < 0) {
            mp[downY[-s[i].se - 1]] = -s[i].se - 1;
        } else if (s[i].se >= m) {
            mp.erase(downY[s[i].se - m]);
        } else if (!mp.empty() && mp.upper_bound(y[s[i].se]) != mp.begin() &&
                   y[s[i].se] <= upY[(--mp.upper_bound(y[s[i].se]))->se]) {
            killed_by[(--mp.upper_bound(y[s[i].se]))->se] =
                min(killed_by[(--mp.upper_bound(y[s[i].se]))->se], s[i].se);
        }
    }

    Vi ans(m);
    fill(all(ans), -1);
    forn(i, n) {
        if (killed_by[i] < INT_MAX) {
            ans[killed_by[i]] = i + 1;
        }
    }
    forn(i, m) {
        cout << ans[i] << ' ';
    }

    return 0;
}
```