Технокубок

Технокубок 2017: Условия и разбор задач 1 отборочного раунда

727A - Превращение: из A в B

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

У Василия есть число a, которое он хочет превратить в число b. Для этого он может производить два типа операций:

- умножить имеющееся у него число на 2 (то есть заменить число x числом $2 \cdot x$);
- приписать к имеющемуся у него числу цифру 1 справа (то есть заменить число x числом $10 \cdot x + 1$).

Вам надо помочь Василию получить из числа a число b с помощью описанных операций, либо сообщить, что это невозможно.

Обратите внимание, что в этой задаче не требуется минимизировать количество операций. Достаточно найти любой из способов получить из числа a число b.

Входные данные

В первой строке записаны два целых положительных числа a и b ($1 \le a < b \le 109$) — число, которое есть у Василия, и число, которое он хочет получить.

Выходные данные

Если получить число *b* из числа *a* невозможно, выведите «NO» (без кавычек).

В противном случае в первую строку выведите «YES» (без кавычек). Во вторую строку выведите число *k* — количество чисел в последовательности превращений. В третьей строке выведите последовательность превращений *x*1, *x*2, ..., *xk*, причём:

- *x*1 должно быть равно *a*,
- *xk* должно быть равно *b*,
- число *xi* должно быть получено с помощью одной из двух операций из числа *xi* - 1 (1 < *i* ≤ *k*).

Если ответов несколько, разрешается вывести любой из них.

Примеры

входные данные

2 162

выходные данные

YES

5

2 4 8 81 162

входные данные

4 42

выходные данные

NO

входные данные

100 40021

выходные данные

YES

5

100 200 2001 4002 40021

Решение:

Будем решать задачу в обратную сторону — попытаемся получить из числа *B* число *A*.

Заметим, что если число *B* заканчивается на 1, то последней операцией, которую использовал Василий — приписать к числу справа 1. Поэтому удалим последнюю цифру из *B* и перейдем в новому числу.

Если же последняя цифра чётная — то последней операцией, которую использовал Василий — умножить число на 2. Поэтому поделим *B* пополам и перейдем к новому числу.

Если же *B* заканчивается на нечетную цифру отличную от 1, то ответ «NO».

После того как мы перешли к новому числу, нужно вновь выполнить описанный алгоритм. Если на каком-то шаге мы получили число равное *A*, то мы нашли ответ, а если же мы получили число, меньшее *A*, то ответ «NO».

727B - Сумма чека

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Василий вышел из магазина и ему стало интересно пересчитать сумму в чеке. Чек представляет собой строку, в которой названия покупок и их цены записаны подряд без пробелов. Чек имеет вид «*name*1*price*1*name*2*price*2...*namenpricen*», где *namei* (название *i*-го продукта) — это непустая строка длины не более 10, состоящая из строчных букв латинского алфавита, а *pricei* (цена *i*-го продукта) — это непустая строка, состоящая из точек и цифр. Продукты с одинаковым названием могут иметь разные цены.

Цена каждого продукта записана в следующем формате. Если продукт стоит

целое количество рублей, то копейки не пишутся.

Иначе, после записи количества рублей к цене приписывается точка, за которой следом **ровно двумя цифрами** записаны копейки (если копеек менее 10, то используется лидирующий ноль).

Также, каждые три разряда (от менее значимых к более значимым) в записи рублей разделяются точками. Лишние лидирующие нули недопустимы, запись цены всегда начинается с цифры и заканчивается цифрой.

Например, записи цен:

- «234», «1.544», «149.431.10», «0.99» и «123.05» являются корректными,
- «.333», «3.33.11», «12.00», «.33», «0.1234» и «1.2» не являются корректными.

Напишите программу, которая по содержимому чека найдет суммарную цену всех покупок.

Входные данные

В первой строке содержится непустая строка *s* длины не более 1000 — содержимое чека Василия. Гарантируется, что чек задан в формате, описанном в условии. Гарантируется, что каждая цена в чеке составляет не менее одной копейки и не более 106 рублей.

Выходные данные

Выведите суммарную цену всех покупок **строго в том же формате**, в котором задаются цены покупок во входных данных.

Примеры

входные данные

chipsy48.32televizor12.390

выходные данные

12.438.32

входные данные

a1b2c3.38

выходные данные

6.38

входные данные

aa0.01t0.03

выходные данные

0.04

Решение:

В данной задаче нужно было аккуратно сделать то, что написано в условии. Можно было сначала выделить все последовательности из подряд идущих цифр и точек, которые являлись ценами.

Затем нужно было выделить целое количество рублей из каждой цены и отдельно считать сумму всех целых цен в переменную *r*. То же нужно было делать для копеек из каждой цены и складывать их в переменную *c*.

После обработки всех цен, нужно было перевести копейки в рубли, то есть прибавить к *r* величину *c* / 100 (целая часть от деления *c* на 100), а *c* присвоить значению *c*%100 (остаток от деления *c* на 100). После этого осталось только аккуратно вывести ответ, не забыв, что если $c < 10$, то сначала для копеек нужно вывести 0, а затем *c*, так как количество копеек по условию обязательно должно состоять из двух цифр.

727C - Восстановление массива

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Это интерактивная задача. Вам нужно использовать операцию flush после вывода каждого запроса. Например, в C++ вы должны использовать

функцию fflush(stdout), в Java — использовать System.out.flush(), а в Паскале — flush(output).

В этой задаче вам надо восстановить массив, который заранее вам неизвестен. Вы можете считать, что жюри загадало некоторый массив *a*, про который вам известна только его длина *n*.

Единственное допустимое действие — узнать сумму пары элементов, указав их индексы *i* и *j* (индексы должны быть **различными**). В результате запроса для индексов *i* и *j* вы получите сумму *ai* + *aj*.

Известно, что восстановить весь загаданный массив можно не более чем за *n* запросов. Напишите программу, которая восстановит загаданный жюри массив *a* длины *n* за не более чем *n* запросов на сумму двух элементов (в каждом запросе индексы двух элементов должны быть различны).

Протокол взаимодействия

Каждый тест в этой задаче состоит из одного массива, который ваша программа должна восстановить.

В первой строке входных данных следует целое положительное число *n* ($3 \le n \le 5000$) — длина загаданного массива. В первую очередь ваша программа должна прочитать это число.

Далее ваша программа должна выводить в стандартный вывод запросы на сумму двух элементов массива, либо сообщить о том, что загаданный жюри массив уже найден.

- В случае, если программа осуществляет запрос на сумму, то следует вывести строку вида «? i j» (*i* и *j* — **различные** целые числа от 1 до *n*) — индексы элементов массива, сумму которых ваша программа запрашивает.
- В случае, если программа сообщает восстановленный массив, то следует вывести строку вида «! *a*1 *a*2 ... *an*» (гарантируется, что все *ai* в правильно восстановленном массиве — положительные целые числа и не превосходят 105), где *ai* равно числу, стоящему в массиве в позиции *i*.

Результатом запроса на сравнение является единственное целое число,

равное *ai* + *aj*.

Для массива длины *n* ваша программа должна сделать не более *n* запросов на сумму. Обратите внимание, что вывод строки вида «! *a*1 *a*2 ... *an*» не считается запросом и не учитывается при подсчете их количества.

Не забывайте использовать операцию flush после каждой выведенной строки.

После вывода ответа ваша программа должна завершиться.

Пример

входные данные

5

9

7

9

11

6

выходные данные

? 1 5

? 2 3

? 4 1

? 5 2

? 3 4

! 4 6 1 5 5

Примечание

Вы можете взламывать, задавая тесты следующего вида:

• в первой строке должно быть записано целое число *n* (3 ≤ *n* ≤ 5000) — длина массива,

во второй строке должны быть записаны целые числа *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 105) — элементы загаданного массива.

Решение:

Изначально сделаем три запроса на сумму чисел $a1 + a2 = c1$, $a1 + a3 = c2$ и $a2 + a3 = c3$.

После этого мы получаем систему из трех уравнений с тремя неизвестными $a1$, $a2$, $a3$. После простых вычислений получим, что $a3 = (c3 - c1 + c2) / 2$. После этого легко находятся $a1$ и $a2$. Теперь мы знаем значения $a1$, $a2$, $a3$, потратив на это 3 запроса.

Затем для всех i от 4 до n нужно сделать запрос на сумму $a1 + ai$. Если очередная сумма равна ci, то $ai = ci - a1$ (напомним, что мы уже знаем значение $a1$).

Таким образом можно восстановить весь массив, потратив на это ровно n запросов.

727D - Распределение футболок

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

В качестве сувениров на соревновании по программированию было решено вручить футболки. Всего в типографии были напечатаны футболки шести размеров: S, M, L, XL, XXL, XXXL (размеры перечислены в порядке возрастания). Для каждого размера от S до XXXL вам известно количество футболок такого размера.

Во время регистрации организаторы попросили каждого из n участников указать размер футболки. Если участник колебался между двумя размерами, то он мог указать два соседних — это означает, что ему подойдет футболка любого из двух размеров.

Напишите программу, которая определит, возможно ли из напечатанных в

типографии футболок сделать подарок каждому участнику соревнования. Конечно, каждому участнику должна достаться футболка его размера:

• требуемого размера, если указан один размер;

• любого из двух размеров, если указаны два соседних размера.

В случае положительного ответа программа должна найти любой из вариантов раздачи футболок.

Входные данные

В первой строке входных данных содержится шесть целых неотрицательных чисел — количество футболок с размерами S, M, L, XL, XXL, XXXL, соответственно. Суммарное количество напечатанных футболок не превосходит 100 000.

Во второй строке содержится целое положительное число n ($1 \le n \le 100\,000$) — количество участников соревнований.

В следующих n строках содержатся указания участников — в i-й из них содержатся размеры, указанные i-м участником. Если участник указал только один размер, строка совпадает с его названием. Если же участник указал два размера, то они заданы своими названиями через запятую без разделения пробелами. В этом случае сначала записан меньший размер, затем — больший. Гарантируется, что в таком случае эти размеры соседние.

Выходные данные

Если ответа не существует выведите «NO» (без кавычек).

В противном случае, в первой строке выведите «YES» (без кавычек). В следующих n строках выведите размеры футболок, которые получат участники. Порядок участников должен совпадать с тем порядком, который задан во входных данных.

Если решений несколько, выведите любое.

Примеры

входные данные

0 1 0 1 1 0

3

XL

S,M

XL,XXL

выходные данные

YES

XL

M

XXL

входные данные

1 1 2 0 1 1

5

S

M

S,M

XXL,XXXL

XL,XXL

выходные данные

NO

Решение:

Пусть в массиве *cnt* хранится сколько в типографии есть футболок каждого из размеров.

Изначально раздадим футболки тем, кто точно хочет футболку одного размера, уменьшая при этом соответствующее значение в массиве *cnt*. Если в какой-то момент футболки не хватило, то ответа не существует.

Теперь осталось раздать футболки тем, кто хочет футболку одного из двух размеров. Поступим жадным образом. Раздадим по максимуму футболки размера S тем, кто хочет их или футболки размера M. После этого перейдем к

футболкам размера M и сначала раздадим их по максимуму тем, кто хочет футболки размера S или M, но кому не хватило футболок S, а затем, если остались футболки размера M, раздадим их тем, кто хочет их или футболки размера L. Аналогичным образом раздадим футболки оставшихся размеров.

Если после всех операций с футболками у кого-то футболки не оказалось, значит ответа не существует, в противном случае, ответ найден.

727E - Игры на диске

ограничение по времени на тест: 4 секунды

ограничение по памяти на тест: 512 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

У Толи было n компьютерных игр, и он решил записать их на один диск. После этого он решил написать маркером названия всех своих игр на этом диске по кругу **по часовой стрелке** друг за другом. Названия всех игр были различные, а длина каждого названия была ровно k. Написанные названия на диске не перекрываются между собой.

После того, как Толя написал названия всех игр, на диске получилась циклическая строка длины $n \cdot k$.

Прошло несколько лет и Толя уже и забыл, какие игры записаны на его диске. Он помнит, что всего в то время было g популярных игр, и на его диске могут быть только лишь эти игры, причем каждая из g игр может быть записана на диске **не более одного раза**.

Перед вами стоит задача восстановить любой корректный список игр, которые Толя мог записать на свой диск.

Входные данные

В первой строке следует два целых положительных числа n и k ($1 \le n \le 105$,

$1 \le k \le 105$) — количество игр, которые были у Толи и длина названий этих игр.

Во второй строке следует строка, состоящая из строчных букв латинского алфавита — строка, написанная на диске, разорванная в произвольном месте. Длина этой строки равна $n \cdot k$. Гарантируется, что длина строки не превосходит 106.

В третьей строке следует целое положительное число g ($n \le g \le 105$) — количество популярных игр, которые могут быть записаны на диске. Гарантируется, что суммарная длина всех названий популярных игр не превосходит $2 \cdot 106$.

В следующих g строках следует по одному названию популярных игр. Длина каждого из названий равна k. Строки состоят из строчных букв латинского алфавита. Гарантируется, что все названия популярных игр различны.

Выходные данные

Если ответа не существует, выведите «NO» (без кавычек).

В противном случае, выведите в первую строку «YES» (без кавычек). Во вторую строку выведите n целых чисел — номера популярных игр, записанных на диске Толи. Выводить игры нужно в порядке их записи на диске, то есть **по часовой стрелке**, при этом начинать вывод можно **с любой игры**. Помните, что каждая популярная игра могла быть записана на диск не более одного раза. Если ответов несколько, разрешается вывести любой из них.

Примеры

входные данные

3 1

abc

4

b

a

c

d

выходные данные

YES

2 1 3

входные данные

4 2

aabbccdd

4

dd

ab

bc

cd

выходные данные

NO

Решение:

С помощью алгоритма Ахо-Корасика построим суффиксное дерево на множестве названий игр так, что в вершине дерева, соответствующей названию некоторой игры, (вершине на глубине *k*) будем хранить номер этой игры.

Построенный бор позволяет приписывать к некоторой строке символы один за другим и определять вершину в нем, соответствующую наидлиннейшему префиксу из всех префиксов названий игр, совпадающему с суффиксом нашей строки. Если длина этого префикса равна *k*, то суффикс совпадает с некоторым названием игры.

Запишем строку из входных данных дважды и посчитаем *idxi* — индекс игры, название которой совпадает с подстрокой удвоенной строки с индекса *i* - *k* + 1 по индекс *i* включительно (если такой нет, то - 1).

Теперь остается перебрать индекс символа, который является последним в записи названия некоторой игры на диск. Очевидно, этот индекс можно

перебирать от 0 до *k* - 1. С фиксированным индексом *f* достаточно проверить, что все названия с последними символами в индексах $f + ik \mod nk$ для $0 \le i < n$ являются различными (для этого смотрим, что среди *idx*(*f* + *ik*)%*nk* + *nk* нет - 1 и все они различны). Если это выполняется — выводим YES и легко восстанавливаем ответ. Если ни для одного *f* условия не выполнились, выводим NO.

Асимптотика решения — $O(nk + \sum |t_i|)$

727F - Задачи Поликарпа

ограничение по времени на тест: 2 секунды

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Поликарп — опытный участник соревнований по программированию Codehorses. Теперь он решил попробовать себя в качестве автора задач.

Он отослал координатору раундов набор из *n* задач. Каждая задача характеризуется своим качеством, качество *i*-й задачи равно *ai* (*ai* может быть положительно, отрицательно или равно нулю). Задачи отсортированы по предполагаемой сложности, которая никак не связана с качеством. Таким образом, самая простая задача имеет номер 1, а самая сложная — номер *n*.

В настоящий момент настроение координатора равно *q*. Известно, что после чтения очередной задачи его настроение изменяется на качество этой задачи, то есть после того, как координатор прочитает задачу с качеством *b*, к его настроению добавляется величина *b*. Координатор всегда читает задачи подряд от самой простой к самой сложной, порядок чтения задач изменять нельзя.

Если в какой-то момент текущее настроение координатора становится

отрицательным, то он немедленно прекращает чтение и полностью отклоняет весь комплект задач.

Поликарп хочет выбросить минимальное количество задач так, чтобы настроение координатора всегда было неотрицательным. Так как Поликарп не знает точно текущее настроения координатора, то у него есть m гипотез вида «текущее настроение координатора $q = bi$».

Для каждой из m гипотез найдите минимальное количество задач, которое надо удалить из комплекта, чтобы при чтении оставшихся задач от самой простой к самой сложной настроение координатора всегда было больше или равно 0.

Входные данные

В первой строке входных данных записаны два целых числа n и m ($1 \le n \le 750$, $1 \le m \le 200\,000$) — количество задач в комплекте и количество гипотез о настроении координатора соответственно.

Во второй строке записаны n целых чисел $a1, a2, ..., an$ ($-109 \le ai \le 109$) — качество задач в комплекте в порядке увеличения сложности.

В третьей строке содержится m целых чисел $b1, b2, ..., bm$ ($0 \le bi \le 1015$) — возможные значения текущего настроения координатора q.

Выходные данные

Выведите m строк, в i-й из них выведите единственное целое число — ответ на задачу для $q = bi$.

Пример

входные данные

6 3

8 -5 -4 1 -7 4

0 7 3

выходные данные

2

0

1

Решение:

Для начала решим задачу для одного значения *Q*. Нетрудно показать, что оптимальным поведением является следующее: добавляем в множество оставленных задач очередную задачу качества *ai*; пока значение настроения (сумма качеств и *Q*) является отрицательным, удаляем из множества оставленных задач задачу с наихудшим качеством. Качество такой задачи обязательно будет отрицательным, поэтому мы не испортим значения настроения на предыдущих задач. Такое моделирование просто осуществляется с помощью структур std::set или std::priority_queue.

Соображение выше позволяет нам отвечать на запрос за $O(n \log n)$, однако $O(mn \log n)$ не укладывается в ограничение по времени. Поэтому нужно заметить, что при увеличении *Q* количество удаленных задач не увеличивается, а возможных таких количеств всего *n*. Таким образом, на задачу нужно взглянуть с обратной стороны: для $0 \le x \le n$ посчитать, какое наименьшее значение может иметь *Q* так, что количество удаленных задач не превосходит *x*. Эта задача просто решается для каждого *x* с помощью бинпоиска за $O(n \log n \log MAXQ)$, в сумме по всем *x* получим $O(n^2 \log n \log MAXQ)$. Если еще и учесть, что нас интересуют только *m* значений *Q*, то бинпоиск осуществлять можно только по ним и получить $O(n^2 \log n \log m)$

По сохраненным значениям для каждого ответа *x* остается только найти первый ответ, наименьшее значение *Q* для которого не больше значения *Q* в запросе. Это можно делать наивно за *O*(*n*) или же с помощью бинарного поиска за $O(\log n)$ (значения *Q* для ответов не возрастают), получая *O*(*mn*) или $O(m \log n)$ в сумме.

Наилучшая асимптотика составит $O(n^2 \log n \log m + m \log n)$, однако решения за $O(n^2 \log n \log MAXQ + mn)$ тоже проходят все тесты.