Содержание

1 Задача A. Ancestor. Предок

Имя входного файла:	`ancestor.in`
Имя выходного файла:	`ancestor.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напишите программу, которая для двух вершин дерева определяет, является ли одна из них предком другой.

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 100\,000$) — количество вершин в дереве. Во второй строке находится n чисел. При этом i-ое число второй строки определяет непосредственного родителя вершины с номером i. Если номер родителя равен нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leqslant m \leqslant 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leqslant a, b \leqslant n$).

Формат выходных данных

Для каждого из m запросов выведите на отдельной строке число `1`, если вершина a является одним из предков вершины b, и `0` в противном случае.

Пример

ancestor.in	ancestor.out
6 0 1 1 2 3 3 5 4 1 1 4 3 6 2 6 6 5	0 1 1 0 0

2 Задача B. Число

Имя входного файла:	`number.in`
Имя выходного файла:	`number.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вася написал на длинной полоске бумаги большое число и решил похвастаться своему старшему брату Пете этим достижением. Но только он вышел из комнаты, чтобы позвать брата, как его сестра Катя вбежала в комнату и разрезала полоску бумаги на несколько частей. В результате на каждой части оказалось одна или несколько идущих подряд цифр.

Теперь Вася не может вспомнить, какое именно число он написал. Только помнит, что оно было очень большое. Чтобы утешить младшего брата, Петя решил выяснить, какое максимальное число могло быть написано на полоске бумаги перед разрезанием. Помогите ему!

Формат входных данных

Входной файл содержит одну или более строк, каждая из которых содержит последовательность цифр. Количество строк во входном файле не превышает 100, каждая строка содержит от 1 до 100 цифр. Гарантируется, что хотя бы в одной строке первая цифра отлична от нуля.

Формат выходных данных

Выведите в выходной файл одну строку — максимальное число, которое могло быть написано на полоске перед разрезанием.

Примеры

number.in	number.out
2 20 004 66	66220004
3	3

3 Задача C. Get the Duck to the Sink

Имя входного файла:	`getduck.in`
Имя выходного файла:	`getduck.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 Mebibytes

Как-то профессор Налейпиво заметил, что один из студентов на его лекции уделяет слишком много внимания мобильному телефону. Подкравшись сзади (а несмотря на большие размеры, профессор Налейпиво умеет незаметно подкрадываться), профессор обнаружил смягчающее вину студента обстоятельство — тот не отправлял SMS-ки, а увлечённо играл в следующую игру.

Есть поле размером $N \times M$ ячеек и несколько (возможно ноль) стен между ячейками. Одна из ячеек является *стоком*, в то время как одна из оставшихся занята *уткой*. Ваша задача привести *утку* в *сток*. Единственный способ передвижения *утки*, доступный вам — это *скольжение*, что подразумевает, что вы можете толкнуть *утку* в одном из четырёх направлений, и она будет *скользить* в нем, пока не упрется в стену. Вы не можете толкнуть *утку* снова, пока она не остановится. Уровень считается пройденным, если *утка* останавливается в *стоке*. Если *утка* просто *проскользнула* через *сток*, не остановившись, уровень не считается пройденным.

Профессор остановил лекцию и попросил студента создать несколько своих уровней. Он расчертил на доске уровень, нанёс стены, поместил *сток*, и теперь студенту надо разместить где-то *утку*. Профессор выбрал несколько мест, куда бы он хотел поместить её, и предложил студенту описать алгоритм прохождения уровня. Студент быстро понял, что это ловушка — среди них есть такие, начав игру из которых, довести *утку* до *стока* невозможно. А сумеете ли это сделать Вы?

Ваша задача — имея размеры поля, позицию стен и *стока*, а также выбранные позиции для *утки*, найти среди выбранных позиций такие, начав игру из которых пройти уровень возможно.

Формат входных данных

Первая строка содержит два числа N и M — размеры поля.

Далее следует $2N + 1$ строк, каждая по $2M + 1$ символов, где $2k$-ый символ $2i$-ой стоки либо пробел, если ячейка пуста, либо `S` если ячейка содержит сток либо `D` если ячейка входит в список выбранных для размещения утки.

$(2k + 1)$-ый символ $2i$-ой строки либо пробел, если $k > 0$ и $k < M$ и нет стены между ячейками (k,i) и $(k + 1,i)$; либо `|` в противном случае.

$2k$-ый символ $(2i + 1)$-ой строки либо пробел, если $i > 0$ и $i < N$ и нет стены между ячейками (k,i) и $(k,i + 1)$; либо `-` в противном случае.

$(2k + 1)$-ый символ $(2i + 1)$-ой строки всегда `+`.

$1 \leqslant N \leqslant 1000$

$1 \leqslant M \leqslant 1000$

Формат выходных данных

Выведите поле из входного файла, заменив буквы `D` на пробел в ячейках, начиная с которых, нельзя пройти уровень.

Пример

getduck.in	getduck.out
5 5 +-+-+-+-+-+ | | + +-+ + + + | S D D| + + + + + + | D | | + + + + + + | | | | + + + +-+ + | | +-+-+-+-+-+	+-+-+-+-+-+ | | + +-+ + + + | S D | + + + + + + | D | | + + + + + + | | | | + + + +-+ + | | +-+-+-+-+-+

4 Задача D. incrementator

Имя входного файла:	`incrementator.in`
Имя выходного файла:	`incrementator.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Ваша задача — написать программу, моделирующую простое устройство, которое умеет прибавлять целые значения к целочисленным переменным.

Формат входных данных

Входной файл состоит из одной или нескольких строк, описывающих операции. Строка состоит из названия переменной и числа, которое к этой переменной надо добавить. Все числа не превосходят 100 по абсолютной величине. Изначально все переменные равны нулю. Названия переменных состоят из не более чем 100 000 маленьких латинских букв. Размер входного файла не превосходит 2 мегабайта.

Формат выходных данных

Для каждой операции выведите на отдельной строке значение соответствующей переменной после выполнения операции.

Примеры

incrementator.in	incrementator.out
a 2 b 3 a -1 c 4 b 17 xyz 23	2 3 1 4 20 23

Замечание

Пожалуйста, не используйте встроенные в язык `map`, `set`.

5 Задача E. CHM

Имя входного файла:	`snm.in`
Имя выходного файла:	`snm.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Ваша задача — реализовать **Persistent Disjoint-Set-Union**. Что это значит?
Про **Disjoint-Set-Union**:
Изначально у вас есть n элементов. Нужно научиться отвечать на 2 типа запросов.

- `+ a b` — объединить множества, в которых лежат вершины a и b
- `? a b` — сказать, лежат ли вершины a и b сейчас в одном множестве

Про **Persistent**:
Теперь у нас будет несколько копий (версий) структуры данных **Disjoint-Set-Union**.
Запросы будут выглядеть так:

- `+ i a b` — запрос к i-й структуре, объединить множества, в которых лежат вершины a и b. При этом i-я структура остается не изменной, создается новая версия, ей присваивается новый номер (какой? читайте дальше)
- `? i a b` — запрос к i-й структуре, сказать, лежат ли вершины a и b сейчас в одном множестве

Формат входных данных

На первой строке 2 числа N ($1 \leqslant N \leqslant 10^5$) и K ($0 \leqslant K \leqslant 10^5$) — число элементов и число запросов. Изначально все элементы находятся в различных множествах. Эта изначальная копия (версия) структуры имеет номер 0.

Далее следуют K строк, на каждой описание очередного запроса. Формат запросов описан выше. Запросы нумеруются целыми числами от 1 до K.

При обработки j-го запроса вида `+ i a b`, новая версия получит номер j.

Формат выходных данных

Для каждого запроса вида `? i a b` на отдельной строке нужно вывести `YES` или `NO`.

Пример

snm.in	snm.out
4 7 + 0 1 2 ? 0 1 2 ? 1 1 2 + 1 2 3 ? 4 3 1 ? 0 4 4 ? 4 1 4	NO YES YES YES NO

6 Задача F. Сумма не без разнообразия

Имя входного файла:	`threemax.in`
Имя выходного файла:	`threemax.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Задана последовательность целых чисел $A_1, A_2, \ldots, A_N$.

Необходимо выбрать из нее подпоследовательность из подряд стоящих чисел $A_i, A_{i+1}, \ldots, A_j$ так, чтобы она содержала не менее K различных чисел, и сумма $S = A_i + A_{i+1} + \ldots + A_j$ была максимальной.

Формат входных данных

Первая строка ввода содержит целые числа N и K ($1 \leqslant K \leqslant N \leqslant 200\,000$).

Вторая строка содержит N целых чисел $A_1, A_2, \ldots, A_N$ ($|A_i| \leqslant 1\,000\,000\,000$).

Формат выходных данных

В первой строке необходимо вывести максимальное возможное значение суммы S. Во второй строке выведите индексы первого и последнего элементов найденной оптимальной подпоследовательности. Если существует несколько решений, подойдет любое из них.

Если не существует подпоследовательностей, удовлетворяющих решению задачи, выведите одну строку со словом "`IMPOSSIBLE`" (без кавычек).

Примеры

threemax.in	threemax.out
7 3 -99 1 2 -100 3 2 3	-89 2 7
3 2 1 1 1	IMPOSSIBLE

7 Задача G. Анти-Фибоначчи

Имя входного файла:	`antifib.in`
Имя выходного файла:	`antifib.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Л¨ше надоели числа Фибоначчи. Всю последнюю неделю, когда он приходил на урок математики или информатики, учителя рассказывали что-то про числа Фибоначчи и задавали на дом задачки про них.

На этой неделе домашнее задание у Л¨ши — написать программу, которая по заданному целому положительному числу N находит количество способов разбить N на положительные целые слагаемые. Способы, отличающиеся лишь порядком слагаемых, считаются одинаковыми. К примеру, для $N = 4$ это количество способов — 5:

$$\begin{aligned} N &= 4 \\ &= 3 + 1 \\ &= 2 + 2 \\ &= 2 + 1 + 1 \\ &= 1 + 1 + 1 + 1 \end{aligned}$$

Поскольку Л¨ше не нравятся числа Фибоначчи, он решил написать программу, которая считает только такие разбиения, в которых среди слагаемых нет чисел Фибоначчи. Более того, разбиения, в которых количество слагаемых является числом Фибоначчи, Л¨ша тоже решил не считать.

Помогите Л¨ше написать такую программу.

Формат входных данных

В первой строке входного файла записано целое число N ($1 \leqslant N \leqslant 50$).

Формат выходных данных

Выведите в выходной файл одно число — количество таких разбиений N на положительные целые слагаемые, что ни какое-либо из слагаемых, ни их количество не являются числами Фибоначчи.

Пример

antifib.in	antifib.out
4	0