XXVI Всеукраинская олимпиада по информатике

Первый тур

1. Календарь

TL = 100 мс

Ученые-археологи планеты Олимпия нашли две пещеры с признаками пребывания доисторических племен. Их внимание привлекли N различных слов, начертанных на стене в каждой из пещер. Интересно, что эти слова в обеих пещерах оказались одинаковыми, но выписанными в разной последовательности. Ученые выяснили:

1. Начертанные слова — это названия месяцев года, которые перечислены в порядке наступления у соответствующего племени.
2. Год у племен был разбит на N месяцев равной продолжительности, а дни начала месяцев совпадали.

Однако, ученые так и не определили, в какой день начинался год у каждого из племен.

Задание

Напишите программу **calendar**, которая по данным о последовательности названий месяцев в обеих пещерах найдет наибольшее количество месяцев, которые могли бы иметь одинаковые названия у обоих племен, учитывая, что год у племен мог начинаться в разные моменты времени. Для упрощения анализа ученые сопоставили каждому из названий месяцев свой номер — натуральное число от 1 до N.

Входные данные

Входной файл **calendar.dat** состоит из трех строк. В первой строке содержится натуральное число N ($2 \le N \le 10^5$) — количество названий месяцев, начертанных на стене каждой из пещер. Вторая строка содержит N различных натуральных чисел, каждое из которых не превышает N, — номера слов в порядке, в котором они начертаны в *первой* пещере. Третья строка также содержит N различных натуральных чисел, каждое из которых не превышает N, — номера слов в порядке, в котором они начертаны во *второй* пещере.

Выходные данные

Выходной файл **calendar.sol** должен содержать единственное число — наибольшее количество месяцев, которые могли бы называться одинаково у обоих племен.

Оценивание

1. 20 % баллов: $2 \le N \le 5$.
2. 20 % баллов: $5 < N \le 150$.
3. 20 % баллов: $150 < N \le 3000$.
4. 40 % баллов: $3000 < N \le 10^5$.

Примеры входных и выходных данных

calendar.dat	calendar.sol
4 2 4 3 1 4 2 1 3	2
3 3 2 1 1 2 3	1

Пояснение к первому примеру

Если год у второго племени начинается, например, на месяц позже, чем у первого, то два месяца имеют у племен одинаковые названия (номер 1 и 4):

Племя 1:	3	1	2	4	3	1	2	4	3	1	2	4
Племя 2:	2	1	3	4	2	1	3	4	2	1	3	4

Никакая другая комбинация не приводит к совпадению большего количества названий месяцев.

Пояснение ко второму примеру

Независимо от того, когда именно у племен начинается год, одинаковое название всегда будет иметь только один месяц.

2. Мутация

TL = 300 мс

Ученые-генетики планеты Олимпия опять проводят эксперименты с ДНК примитивных организмов. Геном организма — это последовательность генов, каждый из которых можно закодировать одним натуральным числом. Гены, которые кодируются одними и теми же числами, считаются одинаковыми, и наоборот, гены, которые кодируются разными числами, считаются разными.

Ученые уже вывели некоторый примитивный организм и хотят модифицировать ему геном таким образом, чтобы получить идеальный организм. Они считают, что в дальнейшем это поможет найти лекарства от многих болезней.

Организм считается идеальным, если любые два одинаковых гена либо стоят на соседних позициях в геноме, либо между ними есть хотя бы один такой же, как они, ген.

За одну операцию ученые могут выбрать и удалить один или несколько *одинаковых* генов из генома организма, после чего вставить их обратно в геном, но, возможно, на другие позиции. Поскольку каждая такая операция ослабляет организм, ученые хотят достичь своей цели, выполнив при этом как можно меньше операций.

Задание

Напишите программу **mutation**, которая по заданному представлению генома определит наименьшее количество операций, необходимое для получения идеального организма.

Входные данные

Первая строка входного файла **mutation.dat** содержит целое число N ($1 \le N \le 10^5$) — количество генов в геноме примитивного организма. В следующей строке записано N натуральных чисел, каждое из которых не превышает N, — последовательность генов в геноме.

Выходные данные

Выходной файл **mutation.sol** должен содержать одно целое число — наименьшее количество операций, за которое ученые смогут получить идеальный организм.

Оценивание

1. 50 % баллов: N не превышает 16.
2. 50 % баллов: нет дополнительных ограничений.

Пример входных и выходных данных

mutation.dat	mutation.sol
9 1 2 1 3 1 3 2 4 5	2

Пояснение. Ниже показана одна из возможных последовательностей выполнения операций. Жирным выделены гены, которые будут перемещены после выполнения следующей операции:

1 **2** 1 3 1 3 **2** 4 5 → **1 1** 3 **1** 3 2 2 4 5 → 1 1 1 3 3 2 2 4 5

3. Космические камешки

TL = 750 мс

Главной драгоценностью на планете Олимпия являются камешки, которые иногда падают на поверхность планеты из космоса. Чем тяжелее камешек, тем он ценнее. Чтобы обеспечить функционирование планетарных учреждений, правительство время от времени собирает с городов Олимпии налог. Из каждого из M городов в столицу привозят по одному камешку. Министр выбирает среди всех камешков самый тяжелый и берет его как налог. Оставшиеся $M - 1$ камешков отвозят назад в города, откуда их привезли. Чтобы сэкономить, каждый город всегда везет в столицу самый легкий из всех драгоценных камней, которые на данный момент имеет в своей сокровищнице.

Задание

Напишите программу **stones**, которая, зная, в каком порядке в города падали камешки из космоса и массы этих камешков, для каждого события сбора налога определит, камешек какого веса правительство взяло как налог.

Входные данные

В первой строке входного файла **stones.dat** записаны два числа: количество событий N и количество городов M, $2 \le M < N \le 2 \cdot 10^5$. Каждое событие бывает одного из двух типов: либо в некотором городе падает камешек (тип 1), либо правительство собирает налог (тип 2). Следующие N строк файла содержат описание соответствующих событий в тому порядке, в котором они происходили. Первое число строки, которая задает событие, — это тип события (1 или 2).

1. Если первое число строки 1, то после него строка содержит еще два натуральных числа T и W, где T — номер города, куда упал камешек, $1 \le T \le M$, а W — вес камешка, $1 \le W < 10^9$.
2. Если первое число строки 2, то оно является единственным в строке.

Считаем, что перед первым событием ни в одном городе не было ни одного камешка. Входные данные гарантируют выполнение таких условий:

1. Массы всех камешков, которые падали на планету, попарно различны.
2. На момент, когда правительство собирает налог, в каждом из M городов есть хотя бы по одному камешку.
3. Правительство собрало налог хотя бы один раз.

Выходные данные

Выходной файл **stones.sol** должен содержать столько строк, сколько событий типа 2 задано во входном файле: для i-го по порядку события типа 2 в i-й строке выходного файла должно быть записано единственное число — масса камешка, взятого правительством как налог при этом событии.

Оценивание

1. 30 % баллов: $2 \le M < N \le 5000$.
2. 30 % баллов: $5000 < N \le 2 \cdot 10^5$, $2 \le M \le 5$.
3. 40 % баллов: $5000 < N \le 2 \cdot 10^5$, $5 < M < N$.

Пример входных и выходных данных

stones.dat	stones.sol
9 2 1 1 9 1 2 3 1 1 4 2 1 1 2 1 2 5 2 2 1 2 1	4 3 5

4. Олимпийские Авиалинии

TL = 900 мс

Император Олимпии решил соединить N городов империи воздушным транспортом. Придворному программисту было приказано написать программу под названием *ОлимпСтрой*, которая упрощает процесс построения карты авиарейсов. Эта программа должна выполнять операции двух типов, каждая из которых зависит от трех параметров A, B и C:

1. Создать новый рейс из города A в город B, перелет по которому занимает C минут.
2. Рассмотреть все авиарейсы, которые начинаются в городе A. Пусть рейс номер i из них заканчивается в городе v_i и занимает t_i минут. Для каждого такого i создать рейс из города B в город v_i, который занимает $t_i + C$ минут.

После выполнения всех операций программа должна для каждого города найти наименьшее время, требуемое для перелета в него из столицы, возможно с пересадками в других городах. Считается, что время посадки, высадки и ожидания в аэропорту равны нулю.

Между двумя городами может быть больше одного рейса, причем перелеты по ним могут занимать разное время. Возможно существование рейсов, которые начинаются и заканчиваются в одном и том же городе. Все рейсы односторонние, то есть наличие рейса из города A в город B еще не гарантирует наличие рейса из города B в город A.

Задание

Напишите программу **olympair**, которая по последовательности из M операций описанных выше типов, выполненных программой *ОлимпСтрой*, для каждого города, кроме столицы, найдет наименьшее время, требуемое для перелета из столицы в этот город.

Входные данные

Первая строка входного файла **olympair.dat** содержит два целых числа N и M ($2 \le N \le 10^5$, $1 \le M \le 10^5$) — количество городов в империи Олимпия и количество операций, выполненных программой *ОлимпСтрой*. В каждой из следующих M строк содержится информация об очередной операции. Первое число в строке равно 1, если выполняемая операция имеет первый тип, и 2, если второй. Далее записаны три целых числа A, B и C ($1 \le A \le N$, $1 \le B \le N$, $-10^8 \le C \le 10^8$), значение которых описано выше. Города нумеруются числами от 1 до N; столица имеет номер 1. Гарантируется, что время перелета по каждому из рейсов будет *строго положительным*.

Выходные данные

Выходной файл **olympair.sol** должен содержать $N - 1$ строку. В i-й из них должно содержаться наименьшее время в минутах, за которое можно долететь из столицы в город с номером $i + 1$. Если до какого-то города добраться невозможно, выведите в соответствующей строке число -1.

Оценивание

Набор тестов состоит из 6 блоков, для которых дополнительно выполняются следующие условия:

1. 15 % баллов: $N \le 2000$, $M \le 4000$.
2. 10 % баллов: выполняются операции только первого типа.
3. 15 % баллов: $C \ge 0$ и если город встретился в качестве A в операции второго типа, то дальше во входном файле он больше не встречается ни в качестве A, ни в качестве B.
4. 15 % баллов: то же самое, что в блоке 3, только на C не накладывается условие неотрицательности.
5. 15 % баллов: $C \ge 0$.
6. 30 % баллов: нет дополнительных ограничений.

Пример входных и выходных данных

olympair.dat	olympair.sol
4 3 1 1 2 10 2 1 3 -9 1 1 3 8	9 8 -1