

Задача А. Парковка

Имя входного файла: `parking.in`
Имя выходного файла: `parking.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

На кольцевой парковке есть n мест пронумерованных от 1 до n . Всего на парковку приезжает n машин в порядке нумерации. У i -й машины известно место p_i , которое она хочет занять. Если машина приезжает на парковку, а её место занято, то она едет далее по кругу и встаёт на первое свободное место.

Формат входных данных

В первой строке входного файла находится число n ($1 \leq n \leq 300\,000$) — размер парковки и число машин. Во второй строке записаны n чисел, i -е из которых p_i ($1 \leq p_i \leq n$) — место, которое хочет занять машина с номером i .

Формат выходных данных

Выведите n чисел: i -е число — номер парковочного места, которое было занято машиной с номером i .

Пример

parking.in	parking.out
3 2 2 2	2 3 1

Задача В. Подсчет опыта

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В очередной онлайн игре игроки, как обычно, сражаются с монстрами и набирают опыт. Для того, чтобы сражаться с монстрами, они объединяются в кланы. После победы над монстром, всем участникам клана, победившего его, добавляется одинаковое число единиц опыта. Особенностью этой игры является то, что кланы никогда не распадаются и из клана нельзя выйти. Единственная доступная операция — объединение двух кланов в один.

Поскольку игроков стало уже много, вам поручили написать систему учета текущего опыта игроков.

Формат входных данных

В первой строке входного файла содержатся числа n ($1 \leq n \leq 200000$) и m ($1 \leq m \leq 200000$) — число зарегистрированных игроков и число запросов.

В следующих m строках содержатся описания запросов. Запросы бывают трех типов:

- `join X Y` — объединить кланы, в которые входят игроки X и Y (если они уже в одном клане, то ничего не меняется).
- `add X V` — добавить V единиц опыта всем участникам клана, в который входит игрок X ($1 \leq V \leq 100$).
- `get X` — вывести текущий опыт игрока X .

Изначально у всех игроков 0 опыта и каждый из них состоит в клане, состоящим из него одного.

Формат выходных данных

Для каждого запроса `get X` выведите текущий опыт игрока X .

Пример

стандартный ввод	стандартный вывод
3 6	150
add 1 100	0
join 1 3	50
add 1 50	
get 1	
get 2	
get 3	

Задача С. Пароль

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вы большой фанат онлайн конкурса по программированию, который называется Звездный Кон-тест(ЗК). Чтобы практиковаться в нем, вы должны скачать приложение Звезда Смерти, войти в него по вашему логину и паролю, и начать соревноваться.

Чтобы избежать атак хакеров на Звезду Смерти, ЗК установило правило для паролей пользова-телей. Теперь первые k символов для каждого пароля должны совпадать с последними k символами. В случае, если кто-то входит по паролю с разными первыми и последними k символами неоднократно, то это считается атакой хакера.

Однако, вы любите свой старый пароль и не хотите изменять много символов. Вам дан ваш старый пароль s и число k . Выведите минимальное количество символов, которые вам необходимо изменить в пароле s , чтобы первые и последние k символов совпадали.

Формат входных данных

Единственная строка содержит число s — ваш старый пароль ($1 \leq |s| \leq 50$), и число k — длина префикса, которая должна равняться суффиксу в новом пароле ($1 \leq k \leq |s|$).

Формат выходных данных

Выведите минимальное число символов, которое необходимо изменить, чтобы префикс и суф-фикс длины k совпадали.

Примеры

стандартный ввод	стандартный вывод
topcoderopen 5	3
пууорпуо 4	0
loool 3	1
arena 5	0

Задача D. Остовное дерево 2

Имя входного файла: `spantree2.in`
Имя выходного файла: `spantree2.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Требуется найти в связном графе остовное дерево минимального веса.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно. Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается тремя натуральными числами b_i , e_i и w_i — номера концов ребра и его вес соответственно ($1 \leq b_i, e_i \leq n$, $0 \leq w_i \leq 100\,000$). $n \leq 20\,000$, $m \leq 100\,000$.

Граф является связным.

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — вес минимального остовного дерева.

Пример

spantree2.in	spantree2.out
4 4 1 2 1 2 3 2 3 4 5 4 1 4	7

Задача Е. Плотное остовное дерево

Имя входного файла: `mindiff.in`
Имя выходного файла: `mindiff.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Требуется найти в графе остовное дерево, в котором разница между весом максимального и минимального ребра минимальна.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно. Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается тремя натуральными числами b_i , e_i и w_i — номера концов ребра и его вес соответственно ($1 \leq b_i, e_i \leq n$, $0 \leq |w_i| \leq 10^9$). $n \leq 1000$, $m \leq 10\,000$.

Формат выходных данных

Если остовное дерево существует, выведите в первой строке выходного файла **YES**, а во второй строке одно целое число — минимальную разность между весом максимального и минимального ребра в остовном дереве.

В противном случае в единственной строке выведите **NO**.

Пример

<code>mindiff.in</code>	<code>mindiff.out</code>
4 5 1 2 1 1 3 2 1 4 1 3 2 2 3 4 2	YES 0

Задача F. Реструктуризация компании

Имя входного файла: `restructure.in`
Имя выходного файла: `restructure.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

В жизни даже самой успешной компании может наступить кризисный период, когда приходится принимать тяжёлое решение о реструктуризации, распускать и объединять отделы, увольнять работников и заниматься прочими неприятными делами. Рассмотрим следующую модель компании.

В Большой Софтверной Компании работают n человек. Каждый человек принадлежит какому-то *отделу*. Исходно каждый человек работает над своим проектом в своём собственном отделе (таким образом, в начале компания состоит из n отделов по одному человеку).

Однако, в жизни компании наступили тяжёлые времена, и руководство было вынуждено нанять кризисного менеджера, который начал переустраивать рабочий процесс для повышения эффективности производства. Обозначим за $team(person)$ команду, в которой работает человек $person$. Кризисный менеджер может принимать решения двух типов:

1. Объединить отделы $team(x)$ и $team(y)$, сформировав из них один большой отдел, содержащий всех сотрудников $team(x)$ и $team(y)$, где x и y ($1 \leq x, y \leq n$) — номера каких-то двух сотрудников компании. Если $team(x)$ совпадает с $team(y)$, ничего делать не требуется.
2. Объединить отделы $team(x), team(x+1), \dots, team(y)$, где x и y ($1 \leq x \leq y \leq n$) — номера каких-то двух сотрудников компании.

При этом кризисный менеджер иногда может интересоваться, работают ли в одном отделе сотрудники x и y ($1 \leq x, y \leq n$).

Помогите кризисному менеджеру, ответив на все его запросы.

Формат входных данных

Первая строка входных данных содержит два целых числа n и q ($1 \leq n \leq 200\,000$, $1 \leq q \leq 500\,000$) — количество сотрудников компании и количество запросов кризисного менеджера.

В последующих q строках находятся запросы кризисного менеджера. Каждый запрос имеет вид $type\ x\ y$, где $type \in \{1, 2, 3\}$. Если $type = 1$ или $type = 2$, то запрос представляет собой решение кризисного менеджера об объединении отделов соответственно первого или второго вида. Если $type = 3$, то требуется определить, работают ли в одном отделе сотрудники x и y . Обратите внимание, что x может равняться y в запросе любого типа.

Формат выходных данных

На каждый запрос типа 3 выведите «YES» или «NO» (без кавычек), в зависимости от того, работают ли в одном отделе соответствующие люди.

Примеры

restructure.in	restructure.out
8 6	NO
3 2 5	YES
1 2 5	YES
3 2 5	
2 4 7	
2 1 2	
3 1 7	

Задача G. Разрезание графа

Имя входного файла: `cutting.in`
Имя выходного файла: `cutting.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Дан неориентированный граф. Над ним в заданном порядке производят операции следующих двух типов:

- **cut** — разрезать граф, то есть удалить из него ребро;
- **ask** — проверить, лежат ли две вершины графа в одной компоненте связности.

Известно, что после выполнения всех операций типа **cut** рёбер в графе не осталось. Найдите результат выполнения каждой из операций типа **ask**.

Формат входных данных

Первая строка входного файла содержит три целых числа, разделённые пробелами — количество вершин графа n , количество рёбер m и количество операций k ($1 \leq n \leq 50\,000$, $0 \leq m \leq 100\,000$, $m \leq k \leq 150\,000$).

Следующие m строк задают рёбра графа; i -я из этих строк содержит два числа u_i и v_i ($1 \leq u_i, v_i \leq n$), разделённые пробелами — номера концов i -го ребра. Вершины нумеруются с единицы; граф не содержит петель и кратных рёбер.

Далее следуют k строк, описывающих операции. Операция типа **cut** задаётся строкой “**cut** u v ” ($1 \leq u, v \leq n$), которая означает, что из графа удаляют ребро между вершинами u и v . Операция типа **ask** задаётся строкой “**ask** u v ” ($1 \leq u, v \leq n$), которая означает, что необходимо узнать, лежат ли в данный момент вершины u и v в одной компоненте связности. Гарантируется, что каждое ребро графа встретится в операциях типа **cut** ровно один раз.

Формат выходных данных

Для каждой операции **ask** во входном файле выведите на отдельной строке слово “**YES**”, если две указанные вершины лежат в одной компоненте связности, и “**NO**” в противном случае. Порядок ответов должен соответствовать порядку операций **ask** во входном файле.

Пример

cutting.in	cutting.out
3 3 7	YES
1 2	YES
2 3	NO
3 1	NO
ask 3 3	
cut 1 2	
ask 1 2	
cut 1 3	
ask 2 1	
cut 2 3	
ask 3 1	

Задача Н. Масло

Имя входного файла: `oil.in`
Имя выходного файла: `oil.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Между пунктами с номерами $1, 2, \dots, N$ ($N \leq 1500$) проложено несколько дорог. Длина каждой дороги известна. По этой системе дорог можно добраться из любого упомянутого пункта в любой другой. Автозаправки расположены только в пунктах. Требуется определить, какое максимальное расстояние без заправки должен быть в состоянии проезжать автомобиль, чтобы без проблем передвигаться между пунктами.

Формат входных данных

В первой строке входного файла находятся числа N и K (количество дорог), $1 \leq N \leq 1500$, $1 \leq K \leq 400\,000$. В следующих K строках указаны пары пунктов, связанных дорогами, и расстояние между ними — целое число километров, не превышающее 10 000.

Формат выходных данных

В выходном файле должно оказаться одно число — длина максимального пробега без дозаправки.

Пример

<code>oil.in</code>	<code>oil.out</code>
3 2 1 2 5 1 3 10	10

Задача I. Нефтяное дело

Имя входного файла: oil.in
Имя выходного файла: oil.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Вам дан неориентированный связный граф из n вершин и m рёбер. Для каждого ребра известна стоимость его удаления в тугриках. У вас есть s тугриков. Вы хотите удалить как можно больше рёбер так, чтобы граф оставался связным, а суммарная стоимость всех удалённых рёбер не превосходила s тугриков.

Формат входных данных

Первая строка входных данных содержит три целых числа n , m и s — количество вершин в графе, количество рёбер в графе и количество тугриков соответственно ($2 \leq n \leq 50\,000$, $1 \leq m \leq 100\,000$, $0 \leq s \leq 10^{18}$). Следующие m строк содержат описания рёбер графа. Каждое описание состоит из трёх целых чисел — номера вершин, которые соединяет данное ребро, и стоимость удаления ребра в тугриках (стоимость не превосходит 10^9). В графе не бывает кратных рёбер и петель.

Формат выходных данных

В первой строке выведите максимальное количество удаляемых рёбер. Во второй строке выведите номера удаляемых рёбер (рёбра нумеруются с единицы в порядке, данном во входном файле).

Пример

oil.in	oil.out
6 7 10 1 2 3 1 3 3 2 3 3 3 4 1 4 5 5 5 6 4 4 6 5	2 1 6