Задача A. Рёбра добавляются, граф растёт

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

В неориентированный граф последовательно добавляются новые рёбра. Изначально граф пустой. После каждого добавления нужно говорить, является ли текущий граф двудольным.

Формат входных данных

На первой строке n — количество вершин, m — количество операций «добавить ребро» ($1 \leqslant n, m \leqslant 300\,000$). Следующие m строк содержат пары чисел от 1 до n — описания добавляемых рёбер.

Формат выходных данных

Выведите в строчку m нулей и единиц: i-й символ должен быть равен единице, если граф, состоящий из первых i рёбер, является двудольным.

Пример

стандартный ввод	*стандартный вывод*
3 3 1 2 2 3 3 1	110

Задача B. Лексикографически минимальная подсеть

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

> Ты думаешь по-японски! Если взялся думать — думай по-немецки!
>
> Neon Genesis Evangelion

Рей Аянами изучает сети из генетически модифицированных деревьев. Сетью называется набор деревьев, соединённых двусторонними биокордами. Биокорды пронумерованы последовательными целыми числами, начиная с единицы. Каждый биокорд соединяет ровно два различных дерева.

Подсеть — это подмножество биокордов сети. Подсеть S устойчива, если:

- между любыми двумя различными деревьями есть путь из биокордов S,
- никакое собственное подмножество S не обладает этим свойством.

Подсети можно сравнивать лексикографически. Чтобы это сделать, следует записать номера биокордов подсети в возрастающем порядке, после чего сравнить лексикографически полученные последовательности чисел. Напомним, что последовательность $a_1, a_2, \ldots, a_p$ *лексикографически меньше* последовательности $b_1, b_2, \ldots, b_p$, если для минимального i, для которого $a_i \neq b_i$, верно $a_i < b_i$.

Рей нужно найти одну конкретную подсеть заданной сети. Во-первых, эта подсеть должна быть устойчивой. Во-вторых, если устойчивых подсетей несколько, нужна та из них, у которой суммарная длина биокордов минимальна. В-третьих, если и таких подсетей несколько, из них нужна лексикографически минимальная.

Рей считает, что справится с этой задачей за несколько минут. А сможете ли вы это сделать?

Формат входных данных

Первая строка входных данных содержит два целых числа n и m — количество генетически модифицированных деревьев и количество двусторонних биокордов между ними ($2 \leqslant n \leqslant 100\,000$, $m \leqslant 100\,000$). Каждая из следующих m строк содержит три целых числа: номера деревьев, соединённых очередным биокордом, и длину этого биокорда. Длины биокордов неотрицательны и не превосходят $100\,000$. Гарантируется, что входные данные таковы, что ответ существует.

Формат выходных данных

Выведите $n - 1$ число: номера биокордов в искомой подсети. Биокорды нумеруются целыми числами от 1 до m в том порядке, в котором они заданы во входных данных.

Пример

стандартный ввод	*стандартный вывод*
4 4 1 2 1 2 3 1 3 4 1 4 1 1	1 2 3

Задача C. Всем чмоки в этом чатике!

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Сегодня Мэри, как программисту социальной сети «Телеграфчик», предстоит реализовать сложную систему управления чатами.

Задача Мэри усложняется тем, что в социальную сеть «Телеграфчик» внедрена продвинутая система шифрования «ZergRus», простая, как всё гениальное. Суть её в том, что в системе хранится одна переменная zerg, которая принимает значения от 0 (включительно) до $p = 10^6 + 3$ (исключая p) и меняется в зависимости от событий в системе.

В социальной сети всего n пользователей ($1 \leqslant n \leqslant 10^5$). В начале дня каждый пользователь оказывается в своём собственном чате, в котором больше никого нет. Переменная zerg в начале дня устанавливается равной 0.

В течение дня происходят события типов:

1. Участник с номером $(i + zerg) \bmod n$ посылает сообщение всем участникам, сидящим с ним в чате (в том числе и себе самому), при этом переменная $zerg$ заменяется на $(30 \cdot zerg + 239) \bmod p$.
2. Происходит слияние чатов, в которых сидят участники с номерами $(i + zerg) \bmod n$ и $(j + zerg) \bmod n$. Если участники и так сидели в одном чате, то ничего не происходит. Если в разных, то чаты объединяются, а переменной $zerg$ присваивается значение $(13 \cdot zerg + 11) \bmod p$.
3. Участник с номером $(i + zerg) \bmod n$ хочет узнать, сколько сообщений он не прочитал, и прочитать их. Если участник прочитал q новых сообщений, то переменной $zerg$ присваивается значение $(100\,500 \cdot zerg + q) \bmod p$.

Вы поможете Мэри реализовать систему, обрабатывающую эти события?

Формат входных данных

В первой строке входных данных записаны натуральные числа n ($1 \leqslant n \leqslant 10^5$) — число пользователей социальной сети. и m ($1 \leqslant m \leqslant 3 \cdot 10^5$) — число событий, произошедших за день. В следующих m строках содержится описание событий. Первое целое число в строке означает тип события t ($1 \leqslant t \leqslant 3$). Если $t = 1$, далее следует число i ($0 \leqslant i < n$), по которому можно вычислить, какой участник послал сообщение. Если $t = 2$, далее следуют числа i и j ($0 \leqslant i, j < n$), по которым можно вычислить номера участников, чаты с которыми должны объединиться. Если $t = 3$, далее следует число i ($0 \leqslant i < n$), по которому можно вычислить номер участника, желающего узнать, сколько у него сообщений, и прочитать их.

Формат выходных данных

Для каждого события типа 3 нужно вывести число непрочитанных сообщений у участника.

Пример

стандартный ввод	*стандартный вывод*	Пояснение
4 10 1 0 1 2 1 1 1 2 3 1 2 1 2 1 3 3 3 2 3 2 3 2	1 1 2	4 10 1 0 1 1 1 2 1 3 3 0 2 0 1 1 1 3 0 2 2 1 3 1

Замечание

Справа указаны номера участников в запросах после декодирования.

Задача D. Соединение точек

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Даны N точек на плоскости. Требуется провести отрезки между некоторыми парами точек таким образом, чтобы, во-первых, из любой данной точки в любую можно было пройти по этим отрезкам, а во-вторых, суммарная длина проведённых отрезков была минимальна.

Формат входных данных

В первой строке входных данных задано число N — количество точек ($1 \leqslant N \leqslant 200$). Следующие N строк содержат по два числа X_i и Y_i каждая через пробел — координаты i-й точки ($-1000 \leqslant X_i, Y_i \leqslant 1000$). Никакие две данные точки не совпадают, никакие три не лежат на одной прямой. Все числа во входных данных целые.

Формат выходных данных

В первой строке выведите L — суммарную длину проведённых отрезков с точностью не менее шести знаков после десятичной точки. Во второй строке выведите K — их количество. В следующих K строках выведите по два числа A_j и B_j через пробел в каждой — номера точек, соединённых j-м отрезком ($1 \leqslant A_j, B_j \leqslant N$, $A_j \neq B_j$). Точки нумеруются с единицы в том порядке, в котором они даны во входных данных. Если ответов с минимальным L несколько, разрешается выводить любой из них.

Примеры

стандартный ввод	*стандартный вывод*
4 0 0 0 1 1 0 1 1	3 3 1 2 2 4 4 3
5 0 0 0 2 1 1 3 0 3 2	7.064495 4 3 1 3 2 3 4 4 5

Задача E. Связность

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Жюри загадало случайный неориентированный граф из n вершин и m рёбер без петель и кратных рёбер. В каждом тесте граф равновероятно выбирается из всех возможных графов с фиксированными n и m до начала проверки вашей программы.

Вашей программе требуется проверить этот граф на связность, исходно зная только n и m. Программа может не более $2 \cdot n$ раз выполнить запрос вида «`?` i» ($1 \leqslant i \leqslant n$). В ответ на такой запрос проверяющая система выдаст одно из двух:

- Положительное целое j ($1 \leqslant j \leqslant n$), означающее, что есть ребро между вершинами i и j, а также что это ребро раньше не было сообщено программе (в том числе в ответе на запрос вида «`?` j»).
- Число -1, означающее, что все рёбра из вершины i программе уже были сообщены.

Протокол взаимодействия

В первой строке входных данных находится целое число T — количество независимых тестовых случаев, которые должна обработать ваша программа. После этого вашей программе требуется T раз провзаимодействовать с проверяющей системой.

Очередное взаимодействие начинается с целых чисел n и m, передаваемых вашей программе на отдельной строке — это количество вершин и рёбер в загаданном графе ($2 \leqslant n \leqslant 10^4$, $1 \leqslant m \leqslant \min(\frac{n(n-1)}{2}, 10^4)$).

После этого вы можете несколько раз вывести на отдельной строке запрос в следующем формате: символ вопроса без кавычек («`?`», ASCII-код 63), за ним пробел, а за ним — целое число i ($1 \leqslant i \leqslant n$) — номер вершины, для которой вы хотите узнать очередное смежное ребро.

В ответ на каждый такой запрос проверяющая система выведет на отдельной строчке либо положительный номер очередной вершины, смежной с i, либо число -1.

Чтобы выдать ответ, выведите на отдельной строке либо «`+`» (плюс, ASCII-код 43), если граф связен, либо «`-`» (минус, ASCII-код 45), если граф не связен.

После выдачи ответа для последнего из T тестовых случаев следует корректно завершить работу программы.

Сумма n по всем тестам за один запуск не превосходит 10^5.

После вывода каждой строки следует очищать буфер вывода, иначе вы получите вердикт `Idleness Limit Exceeded`: это можно сделать, например, вызовом `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Пример

стандартный ввод	*стандартный вывод*
2 5 4 2 1 5 -1 3 -1 5 4 -1	 ? 1 ? 4 ? 1 ? 1 ? 4 ? 4 + ? 5 -

Замечание

Пустые строки добавлены для наглядности, при взаимодействии с проверяющей системой они отсутствуют.

В тесте из примера загаданы два нижеследующих графа. На запрос «`?` i» проверяющая система ответит первым ребром из списка, смежным с вершиной i, которое ещё не было сообщено программе.

1. $n = 5$, $m = 4$, рёбра упорядочены так: 1–2, 1–4, 3–4, 1–5.
2. $n = 5$, $m = 4$, рёбра упорядочены так: 1–4, 1–3, 1–2, 3–4.

Задача F. Разрезание графа

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

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 ввода содержит три целых числа, разделённых пробелами — количество вершин графа n, количество рёбер m и количество операций k ($1 \leqslant n \leqslant 50\,000$, $0 \leqslant m \leqslant 100\,000$, $m \leqslant k \leqslant 150\,000$).

Следующие m строк задают рёбра графа; i-ая из этих строк содержит два числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant n$), 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» ($1 \leqslant u, v \leqslant n$), которая означает, что из графа удаляют ребро между вершинами u и v. Операция типа `ask` задаётся строкой «`ask` u v» ($1 \leqslant u, v \leqslant n$), 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` выведите в 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 вводе.

Пример

стандартный ввод	*стандартный вывод*
3 3 7 1 2 2 3 3 1 ask 3 3 cut 1 2 ask 1 2 cut 1 3 ask 2 1 cut 2 3 ask 3 1	YES YES NO NO

Задача G. Система непересекающихся множеств

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мебибайт

Система непересекающихся множеств — структура данных, хранящая для некоторого множества его разбиение на подмножества в каждый момент времени. Она поддерживает две операции: слияние двух подмножеств и проверку принадлежности двух элементов одному подмножеству.

В этой задаче структура используется так. Дан граф из n вершин, изначально не имеющий рёбер. В граф последовательно предлагаются рёбра для добавления. Каждый раз, когда предлагается ребро между какими-то вершинами u и v веса c, следует проверить, лежат ли на этот момент u и v в одной компоненте связности. Если да, ребро игнорируется. Если нет, ребро добавляется в граф.

Найдите суммарный вес ребёр, добавленных в граф после всех указанных операций.

Формат входных данных

В первой строке заданы через пробел целые числа n и k — количество вершин в графе и количество строк, описывающих рёбра ($1 \leqslant n \leqslant 10^7$, $0 \leqslant k \leqslant 10^5$).

В следующих k строках заданы рёбра. Каждая из них имеет вид $u\ v\ c\ \Delta u\ \Delta v\ \Delta c\ m$ ($0 \leqslant u, v < n$, $0 \leqslant c < 10^9$, $|\Delta u|, |\Delta v|, |\Delta c| < 10^9$, $1 \leqslant m \leqslant 10^7$, все числа в строке целые). Такая строка означает, что последовательно поступают предложения о добавлении в граф m рёбер. Первое из них — ребро между вершинами u и v, имеющее вес c. Второе — ребро между $(u+\Delta u) \bmod n$ и $(v+\Delta v) \bmod n$, имеющее вес $(c+\Delta c) \bmod 10^9$, и так далее. Последнее из этих m рёбер соединяет вершины $(u + (m-1) \cdot \Delta u) \bmod n$ и $(v + (m-1) \cdot \Delta v) \bmod n$, а его вес равен $(c + (m-1) \cdot \Delta c) \bmod 10^9$. Напомним, что $x \bmod y$ — это наименьшее неотрицательное число z такое, что величина $z - x$ делится нацело на y.

Общее количество предлагаемых рёбер, равное сумме чисел m во всех строках, не превосходит 10^7. Среди предлагаемых рёбер могут быть кратные рёбра и петли. Обратите внимание на то, что вершины в графе нумеруются с нуля.

Формат выходных данных

В первой строке выведите одно число — суммарный вес рёбер, добавленных в граф после всех указанных операций.

Примеры

стандартный ввод	*стандартный вывод*
8 3 0 1 1 2 2 0 4 0 1 2 1 1 0 5 0 3 6 9 12 15 2	29
4 1 1 2 1 1 1 1 2	3

Задача N. Игра

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

— Но как он это делает! Он забирается на самую высокую сосну и оттуда планирует.
— Ага. Простите, что планирует?

«День радио»

Девочка Наташа готовит поле для очень интересной игры. В ней примут участие k команд, каждая из которых должна получить в своё распоряжение одно или несколько деревьев и верёвку. При этом у каждой из команд должна быть возможность с помощью верёвки добраться от любого своего дерева до любого другого своего дерева, не используя чужие деревья, но, возможно, используя другие свои деревья как промежуточные. Будем считать, что с помощью верёвки можно перебраться с одного дерева на другое напрямую, если её длина не меньше расстояния между ними.

Длина всех верёвок должна быть одинаковой, чтобы поставить команды в равные условия. Разделите все доступные n деревьев на k наборов так, чтобы необходимая длина верёвок оказалась как можно меньшей.

Формат входных данных

В первой строке записаны целые числа n и k — количество деревьев и команд, соответственно ($1 \leqslant k \leqslant n \leqslant 1000$).

В каждой из следующих n строк записано по два целых числа x_i и y_i — координаты i-го дерева ($-10^4 \leqslant x_i, y_i \leqslant 10^4$).

Формат выходных данных

В первой строке выведите одно вещественное число с шестью или более точными знаками после десятичной точки — минимально возможную длину верёвок. Во второй строке выведите n целых чисел от 1 до k — номера команд, которым следует присвоить соответствующие деревья.

Если решений несколько, выведите любое.

Пример

стандартный ввод	*стандартный вывод*
4 2 1 0 0 1 1 1 0 0	1.00000000 1 2 1 2

Задача I. СНМ и персистентность

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Ваша задача — реализовать **Persistent Disjoint Set Union** (персистентную систему непересекающихся множеств). Что это значит?

Про **Disjoint Set Union**:

Изначально у вас есть n элементов, пронумерованных целыми числами от 1 до n и лежащих каждый в своём множестве. Нужно научиться обрабатывать два типа запросов.

- `+ a b` — объединить множества, в которых лежат элементы a и b
- `? a b` — сказать, лежат ли элементы 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 элементы a и b. При этом i-я структура остаётся неизменной, создаётся новая версия, ей присваивается новый номер (какой? читайте дальше)
- `? i a b` — запрос к i-й структуре: сказать, лежат ли в ней элементы a и b в одном множестве

Формат входных данных

В первой строке заданы два целых числа: число элементов в структуре N и число запросов K ($1 \leqslant N \leqslant 10^5$, $0 \leqslant K \leqslant 10^5$). Изначальная копия (версия) структуры имеет номер 0.

Далее следуют K строк, в каждой из которых записан очередной запрос. Формат запросов описан выше. Запросы нумеруются целыми числами от 1 до K. При обработке j-го запроса, если он имеет вид «`+ i a b`», новая версия получит номер j.

Формат выходных данных

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

Пример

стандартный ввод	*стандартный вывод*
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

Задача J. MST случайных точек

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Даны n различных точек на плоскости. Координаты точек — целые числа от 0 до 30 000 включительно. Точки выбраны *случайно* в следующем смысле: рассмотрим все возможные наборы из n различных точек на плоскости с заданными ограничениями на координаты и выберем из них случайно и равновероятно один набор.

Вы можете провести отрезок между любыми двумя заданными точками. Длина отрезка между точками с координатами (x_1, y_1) и (x_2, y_2) равна $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Будем говорить, что точки a и b *связаны*, если они соединены отрезком, или же существует точка d, которая связана и с a, и с b. Ваша задача — провести отрезки минимальной суммарной длины так, чтобы все точки были связаны.

Формат входных данных

В первой строке ввода задано целое число n ($2 \leqslant n \leqslant 50\,000$). Следующие n строк содержат координаты точек. Гарантируется, что все точки различны. Кроме того, во всех тестах, кроме примера, гарантируется, что точки выбраны случайно, как описано в условии.

Формат выходных данных

В первой строке выведите вещественное число w — суммарную длину отрезков. В следующих $(n-1)$ строках выведите отрезки, по одному на строке. Каждый отрезок следует выводить как два числа от 1 до n, обозначающие номера точек, являющихся концами этого отрезка.

Пусть на самом деле суммарная длина выведенных вами отрезков равна w^*, а суммарная длина отрезков в оптимальном ответе равна w_{opt}. Тогда ваш ответ будет считаться верным, если

$$\max\left(\left|\frac{w}{w^*} - 1\right|, \left|\frac{w^*}{w_{\text{opt}}} - 1\right|\right) < 10^{-12}.$$

Пример

стандартный ввод	*стандартный вывод*
4 0 10 5 6 10 0 0 0	22.02362358924615 1 2 2 3 4 2

Иллюстрация