Задача A. CHM

Имя входного файла:	`snm.in`
Имя выходного файла:	`snm.out`
Ограничение по времени:	2 секунды
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.

Пусть j-й из K запросов имеет вид «`+ i a b`». Тогда новая версия получит номер j. Запросы вида «`? i a b`» не порождают новых структур.

Формат выходных данных

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

Примеры

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

Задача B. Подстроки

Имя входного файла:	`substr.in`
Имя выходного файла:	`substr.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дана строка s. Вам требуется подсчитать количество её различных подстрок. Пустую строку учитывать не следует.

Формат входных данных

В единственной строке входного файла содержится данная строка s, состоящая из строчных латинских букв. Длина строки не превосходит 20 000 символов.

Формат выходных данных

В единственной строке выходного файла выведите единственное число — количество различных подстрок s.

Примеры

substr.in	substr.out
aaaa	4

Задача C. План эвакуации

Имя входного файла:	`evacuate.in`
Имя выходного файла:	`evacuate.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В городе есть муниципальные здания и бомбоубежища, которые были специально построены для эвакуации служащих в случае ядерной войны. Каждое бомбоубежище имеет ограниченную вместительность по количеству людей, которые могут в нем находиться. В идеале все работники из одного муниципального здания должны были бы бежать к ближайшему бомбоубежищу. Однако, в таком случае, некоторые бомбоубежища могли бы переполниться, в то время как остальные остались бы наполовину пустыми.

Чтобы резрешить эту проблему Городской Совет разработал специальный план эвакуации. Вместо того, чтобы каждому служащему индивидульно приписать, в какое бомбоубежище он должен бежать, для каждого муниципального здания определили, сколько служащих из него в какое бомбоубежище должны бежать. Задача индивидального распределения была переложена на внутреннее управление муниципальных зданий.

План эвакуации учитывает количество служащих в каждом здании — каждый служащий должен быть учтен в плане и в каждое бомбоубежище может быть направлено количество служащих, не превосходящее вместимости бомбоубежища.

Городской Совет заявляет, что их план эвакуации оптиматен в том смысле, что суммарное время эвакуации всех служащих города минимально.

Мэр города, находящийся в постоянной конфронтации с Городским Советом, не слишком то верит этому заявлению. Поэтому он нанял Вас в качестве независимого эксперта для проверки плана эвакуации. Ваша задача состоит в том, чтобы либо убедиться в оптимальности плана Городского Совета, либо доказать обратное, представив в качестве доказательства другой план эвакуации с меньшим суммарным временем для эвакуации всех служащих.

Карта города может быть представлена в виде квадратной сетки. Расположение муниципальных зданий и бомбоубежищ задается парой целых чисел, а время эвакуации из муниципального здания с координатами (X_i, Y_i) в бомбоубежище с координатами (P_j, Q_j) составляет $D_{ij} = |X_i - P_j| + |Y_i - Q_j| + 1$ минут.

Формат входных данных

Входной файл содержит описание карты города и плана эвакуации, предложенного Городским Советом. Первая строка входного файла содержит два целых числа N ($1 \leqslant N \leqslant 100$) и M ($1 \leqslant M \leqslant 100$), разделенных пробелом. N — число муниципальных зданий в городе (все они занумерованы числами от 1 до N), M — число бомбоубежищ (все они занумерованы числами от 1 до M).

Последующие N строк содержат описания муниципальных зданий. Каждая строка содержит целые числа X_i, Y_i и B_i, разделенные пробелами, где X_i, Y_i ($-1000 \leqslant X_i, Y_i \leqslant 1000$) — координаты здания, а B_i ($1 \leqslant B_i \leqslant 1000$) — число служащих в здании.

Описание бомбоубежищ содержится в последующих M строках. Каждая строка содержит целые числа P_j, Q_j и C_j, разделенные пробелами, где P_j, Q_j ($-1000 \leqslant P_j, Q_j \leqslant 1000$) — координаты бомбоубежища, а C_j ($1 \leqslant C_j \leqslant 1000$) — вместимость бомбоубежища.

В последующихся N строках содержится описание плана эвакуации. Каждая строка представляет собой описание плана эвакуации для отдельного здания. План эвакуации из i-го здания состоит из M целых чисел E_{ij}, разделенных пробелами. E_{ij} ($0 \leqslant E_{ij} \leqslant 10\,000$) — количество служащих, которые должны эвакуироваться из i-го здания в j-е бомбоубежище.

Гарантируется, что план, заданный во входном файле, корректен.

Формат выходных данных

Если план эвакуации Городского Совета оптимален, то выведите одно слово `OPTIMAL`. В противном случае выведите на первой строке слово `SUBOPTIMAL`, а в последующих N строках выведите

Ваш план эвакуации (более оптимальный) в том же формате, что и во входном файле. Ваш план не обязан быть оптимальным, но должен быть лучше плана Городского Совета.

Примеры

evacuate.in	evacuate.out
3 4 -3 3 5 -2 -2 6 2 2 5 -1 1 3 1 1 4 -2 -2 7 0 -1 3 3 1 1 0 0 0 6 0 0 3 0 2	SUBOPTIMAL 3 0 1 1 0 0 6 0 0 4 0 1
3 4 -3 3 5 -2 -2 6 2 2 5 -1 1 3 1 1 4 -2 -2 7 0 -1 3 3 0 1 1 0 0 6 0 0 4 0 1	OPTIMAL

Задача D. Декомпозиция потока

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

Задан ориентированный граф, каждое ребро которого обладает целочисленной пропускной способностью. Найдите максимальный поток из вершины с номером 1 в вершину с номером n и постройте декомпозицию этого потока.

Формат входных данных

Первая строка входного файла содержит n и m — количество вершин и количество ребер графа ($2 \leqslant n \leqslant 500, 1 \leqslant m \leqslant 10000$). Следующие m строк содержат по три числа: номера вершин, которые соединяет соответствующее ребро графа и его пропускную способность. Пропускные способности не превосходят 10^9.

Формат выходных данных

В первую строку выходного файла выведите одно число — количество путей в декомпозции максимального потока из вершины с номером 1 в вершину с номером n. Следующий строки должны содержать описания элементарых потоков, на который был разбит максимальный. Описание следует выводить в следующем формате: величина потока, количество ребер в пути, вдоль которого течет данный поток и номера ребер в этом пути. Ребра нумеруются с единицы в порядке появления во входном файле.

Примеры

decomposition.in	decomposition.out
4 5 1 2 1 1 3 2 3 2 1 2 4 2 3 4 1	3 1 2 1 4 1 3 2 3 4 1 2 2 5