Задача A. Биномиальная куча

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

Реализуйте **биномиальную** кучу.

Формат входных данных

В первой строке содержится два целых числа: N — общее количество куч и M — количество операций ($1 \leqslant N \leqslant 1000, 1 \leqslant M \leqslant 1\,000\,000$). Изначально все кучи пусты.

Требуется поддерживать следующие операции:

- `0 v a` — добавить элемент со значением v в кучу с номером a. Вновь добавленный элемент имеет уникальный индекс равный порядковому номеру соответствующей операции добавления. Нумерация начинается с единицы.
- `1 a b` — переложить все элементы из кучи с номером a в кучу с номером b. После этой операции куча a становится пустой.
- `2 i` — удалить элемент с индексом i.
- `3 i v` — присвоить элементу с индексом i значение v. Гарантируется, что элемент существует.
- `4 a` — вывести на отдельной строке значение минимального элемента в куче с номером a. Гарантируется, что куча не пуста.
- `5 a` — удалить минимальный элемент из кучи с номером a. Если таковых несколько, то выбирается элемент с минимальным индексом. Гарантируется, что куча не пуста.

Формат выходных данных

Для каждой операции поиска минимального элемента выведите единственное число: значение искомого элемента.

Примеры

binomial.in	binomial.out
3 19 0 1 10 4 1 0 2 5 0 2 7 4 2 3 2 20 4 2 1 2 1 4 1 5 1 4 1 3 2 3 4 1 2 2 4 1 0 1 9 1 1 3 0 3 8 4 3	10 5 7 7 10 3 10 8

Задача B. Левацкая или косая куча

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

Реализуйте **левацкую** или **косую** кучу.

Формат входных данных

В первой строке содержится два целых числа: N — общее количество куч и M — количество операций ($1 \leqslant N \leqslant 1000, 1 \leqslant M \leqslant 1\,000\,000$). Изначально все кучи пусты.

Требуется поддерживать следующие операции:

- `0 v a` — добавить элемент со значением v в кучу с номером a. Вновь добавленный элемент имеет уникальный индекс равный порядковому номеру соответствующей операции добавления. Нумерация начинается с единицы.
- `1 a b` — переложить все элементы из кучи с номером a в кучу с номером b. После этой операции куча a становится пустой.
- `2 i` — удалить элемент с индексом i.
- `3 i v` — присвоить элементу с индексом i значение v. Гарантируется, что элемент существует.
- `4 a` — вывести на отдельной строке значение минимального элемента в куче с номером a. Гарантируется, что куча не пуста.
- `5 a` — удалить минимальный элемент из кучи с номером a. Если таковых несколько, то выбирается элемент с минимальным индексом. Гарантируется, что куча не пуста.

Формат выходных данных

Для каждой операции поиска минимального элемента выведите единственное число: значение искомого элемента.

Примеры

binomial.in	binomial.out
3 19 0 1 10 4 1 0 2 5 0 2 7 4 2 3 2 20 4 2 1 2 1 4 1 5 1 4 1 3 2 3 4 1 2 2 4 1 0 1 9 1 1 3 0 3 8 4 3	10 5 7 7 10 3 10 8

Задача C. Алгоритм двух китайцев

Имя входного файла:	`chinese.in`
Имя выходного файла:	`chinese.out`
Ограничение по времени:	6 секунд
Ограничение по памяти:	256 мегабайт

Вам дан взвешенный ориентированный граф, содержащий n вершин и m рёбер. Найдите минимально возможную сумму весов $n - 1$ ребра, которые нужно оставить в графе, чтобы из вершины с номером 1 по этим ребрам можно было добраться до любой другой вершины.

Формат входных данных

В первой строке даны два целых числа n и m ($1 \leqslant n \leqslant 1\,000$, $0 \leqslant m \leqslant 10\,000$) — количество вершин и ребер в графе.

В следующих m строках даны ребра графа. Ребро описывается тройкой чисел a_i, b_i и w_i ($1 \leqslant a_i, b_i \leqslant n$; $-10^9 \leqslant w_i \leqslant 10^9$) — номер вершины, из которой исходит ребро, номер вершины, в которую входит ребро, и вес ребра.

Формат выходных данных

Если нельзя оставить подмножество ребер так, чтобы из вершины с номером 1 можно было добраться до любой другой, в единственной строке выведите «`NO`».

Иначе, в первой строке выведите «`YES`», а во второй строке выведите минимальную возможную сумму весов ребер, которых необходимо оставить.

Примеры

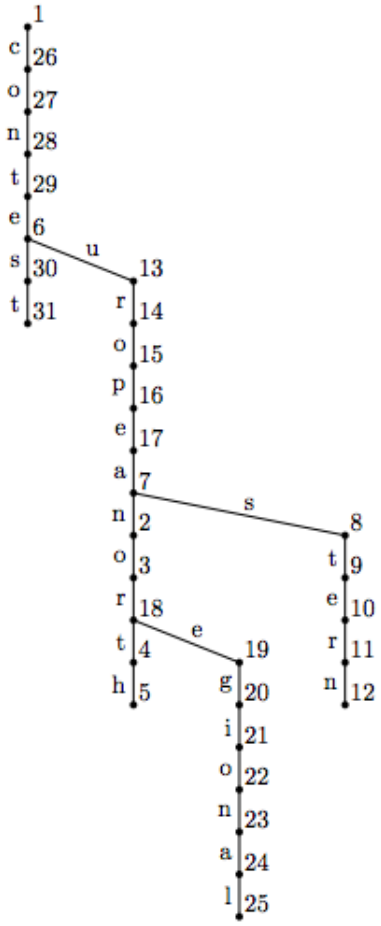
chinese.in	chinese.out
2 1 2 1 10	NO
4 5 1 2 2 1 3 3 1 4 3 2 3 2 2 4 2	YES 6

Задача D. Словарь

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

Петя и Дима работают над новым алгоритмом сжатия данных. Их задача состоит в том, что сжать данный набор слов. Для этого они хотят построить корневое дерево, где на каждом ребре написана ровно одна буква.

Определим для такого дерева словарь, который содержит в точности те слова, которые могут быть получены конкатенацией букв на некотором пути в этом дереве (не обязательно начинающемся в корне), идущим вниз к листу (но не обязательно заканчивающимся в листе).



Ребята хотят построить такое дерево, для которого соотвествующий словарь будет содержать все слова из исходного множества (и, возможно, какие-то еще слова). Среди таких деревьев они хотят выбрать то, которое содержит минимальное количество вершин. Помогите им!

На картинке выше корень дерева имеет номер 1, а путь от вершины 7 до вершины 5 соответствует слову «north», путь от вершины 16 до вершины 12 соответствует слову «eastern», путь от вершины 29 до вершины 2 соответствует слову «european», путь от вершины 3 до вершины 25 соответствует слову «regional», а путь от вершины 1 до вершины 31 соответствует слову “contest”.

Формат входных данных

Первая строка входных данных содержит число слов в множестве n ($1 \leqslant n \leqslant 50$). Следующие n строк содержат различные непустые слова, по одному на строке, каждое из которых состоит из маленьких английских букв. Каждое слово состоит из не более, чем 10 символов.

Формат выходных данных

В первой строке выведите количество вершин в исходном дереве m. В следующих m строках выведите описания вершин дерева. Вершины нумеруются с 1, описание вершины состоит из номера вершины-предка и символа, написанного на ребре, ведущего предка. Для корневой вершины описание должно состоять из единственного числа 0.

Примеры

стандартный ввод	стандартный вывод
5 north eastern european regional contest	31 0 1 c 2 o 3 n 4 t 5 e 6 u 7 r 8 o 9 p 10 e 11 a 12 s 13 t 14 e 15 r 16 n 17 o 18 r 19 t 20 h 19 e 22 g 23 i 24 o 25 n 26 a 27 l 12 n 6 s 30 t

Замечание

Пример соответствует рисунку из условия.

Задача E. Алгоритм двух китайцев. Эпизод второй

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

Вам дан взвешенный ориентированный граф, содержащий n вершин и m рёбер. Найдите минимально возможную сумму весов $n - 1$ ребра, которые нужно оставить в графе, чтобы из вершины с номером 1 по этим ребрам можно было добраться до любой другой вершины.

Формат входных данных

Входные данные содержат описание одного или более тестов.

Каждый тест описывается следующим образом. В первой строке даны два целых числа n и m ($1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 300\,000$) — количество вершин и ребер в графе.

В следующих m строках даны ребра графа. Ребро описывается тройкой чисел a_i, b_i и w_i ($1 \leqslant a_i, b_i \leqslant n$; $-10^5 \leqslant w_i \leqslant 10^9$) — номер вершины, из которой исходит ребро, номер вершины, в которую входит ребро, и вес ребра.

Сумма n по всем тестам не более 100 000. Сумма m по всем тестам не более 300 000.

Формат выходных данных

Для каждого теста выведите или суммарный вес выбранных рёбер, или «NO», если нельзя оставить подмножество рёбер так, чтобы из вершины с номером 1 можно было добраться до любой другой.

Пример

стандартный ввод	стандартный вывод
2 1 2 1 10 4 5 1 2 2 1 3 3 1 4 3 2 3 2 2 4 2	NO 6

Задача F. Хунг Фу

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

Даны два массива целых чисел a и b длины n. Рассмотрим следующую формулу:

$$\sum_{i=1}^{n} \min_{1 \leqslant j \leqslant i} a_i \oplus b_j.$$

Вычислив пару раз значение этой функции, можно заметить, что важен порядок элементов в массиве. Нужно минимизировать значение этого выражения, применив какую-то перестановку к массивам a и b. Другими словами, нужно найти перестановку p, которая минимизирует следующую функцию:

$$F(p) = \sum_{i=1}^{n} \min_{1 \leqslant j \leqslant i} a_{p_i} \oplus b_{p_j}.$$

Найдите и выведите лексикографически минимальную перестановку p, минимизирующую функцию.

Формат входных данных

На первой строке дано одно целое число n: размер массивов ($1 \leqslant n \leqslant 50$).

На второй строке даны n целых чисел a_i: элементы массива a ($0 \leqslant a_i \leqslant 1\,000\,000$).

На третьей строке даны n целых чисел b_i: элементы массива b ($0 \leqslant b_i \leqslant 1\,000\,000$).

Формат выходных данных

На первой строке выведите одно целое число: минимально возможное значение функции.

На второй строке выведите n целых чисел: лексикографически минимальная перестановка p, минимизирующая значение функции.

Примеры

стандартный ввод	стандартный вывод
3 1 2 3 3 2 1	1 2 3 1

Задача G. Персистентная приоритетная очередь

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Требуется реализовать структуру данных, которая хранит мультимножество и умеет изменять любую свою предыдущую версию, выполняя одну из этих операций:

1. Заданы v и x, требуется добавить в множество v элемент со значением x, после чего вывести минимальный элемент в получившемся множестве.

2. Заданы v и u, требуется объединить множества с номерами v и u, после чего вывести минимальный элемент в получившемся множестве.

3. Задано v, требуется вывести минимальный элемент в множестве v, после чего удалить минимальный элемент из множества v. Если множество пустое, то вывести, что множество пустое, и создать новое пустое множество.

Изначально есть одно пустое множество с номером 0. После операции с номером i множество, получаемое во время этой операции, получает номер i.

Формат входных данных

Первая строка содержит число n — количество операций для выполнения.

От вас потребуется отвечать на запросы в онлайне, при этом поддерживая переменную s. Она изначально равна нулю. После каждой операции, она пересчитывается следующим образом через предыдущее значение: если ответ на запрос равен x, то $s = (s_{old} + x) \bmod 239017$. Если же ответом на запрос является слово `empty`, то s не изменяется.

В следующих n строках заданы запросы.

Запросы первого типа описываются строкой `1 a b`, где a и b — неотрицательные целые числа, которые описывают v и x для соответствующего запроса, как $v = (a + s) \bmod i$ и $x = (b + 17s) \bmod (10^9 + 1)$, где i — номер соответствующего запроса.

Запросы второго типа описываются строкой `2 a b`, где a и b — неотрицательные целые числа, которые описывают v и для соответствующего запроса, как $v = (a + s) \bmod i$ и $u = (b + 13s) \bmod i$, где i — номер соответствующего запроса.

Запросы третьего типа описываются строкой `3 a`, где a — неотрицательное целое число, которые описывает v для соответствующего запроса, как $v = (a + s) \bmod i$, где i — номер соответствующего запроса.

Число запросов не превышает 200 000. Гарантируется, что мощность любого созданного мультимножества не превышает 2^{63}.

Формат выходных данных

Требуется вывести ровно n строк, в каждой строке должно находиться неотрицательное целое число либо слово `empty`.

Для запросов первого и второго типа требуется вывести значение минимального элемента в только что созданном множестве, либо слово `empty`, если множество пустое.

Для запросов третьего типа требуется вывести минимальный элемент в множестве, либо слово `empty`, если множество пустое.

Примеры

стандартный ввод	стандартный вывод
9 1 0 2 1 0 999999970 2 2 0 3 0 2 4 4 3 0 3 0 3 0 3 8	2 3 2 2 2 2 2 3 empty