Задача A. Встреченные ранее числа

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

Формат входных данных

Во входной строке записана последовательность из не более, чем 100 000 чисел через пробел. Каждое из чисел не превышает 10^9 по абсолютному значению.

Формат выходных данных

Для каждого числа выведите слово `YES` (в отдельной строке), если это число ранее встречалось в последовательности или `NO`, если не встречалось.

Примеры

metbefore.in	metbefore.out
1 2 3 2 3 4	NO NO NO YES YES NO

Задача B. Коммерческий калькулятор

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

Фирма OISAC выпустила новую версию калькулятора. Этот калькулятор берет с пользователя деньги за совершаемые арифметические операции. Стоимость каждой операции в долларах равна 5% от числа, которое является результатом операции.

На этом калькуляторе требуется вычислить сумму N натуральных чисел (числа известны). Нетрудно заметить, что от того, в каком порядке мы будем складывать эти числа, иногда зависит, в какую сумму денег нам обойдется вычисление суммы чисел (тем самым, оказывается нарушен классический принцип "от перестановки мест слагаемых сумма не меняется":-)).

Например, пусть нам нужно сложить числа 10, 11, 12 и 13. Тогда если мы сначала сложим 10 и 11 (это обойдется нам в \$1.05), потом результат - с 12 (\$1.65), и затем - с 13 (\$2.3), то всего мы заплатим \$ 5, если же сначала отдельно сложить 10 и 11 (\$1.05), потом - 12 и 13 (\$1.25) и, наконец, сложить между собой два полученных числа (\$2.3), то в итоге мы заплатим лишь \$4.6.

Напишите программу, которая будет определять, за какую минимальную сумму денег можно найти сумму данных N чисел.

Формат входных данных

Во входном файле записано число N ($2 \leqslant N \leqslant 10^5$). Далее идет N натуральных чисел, которые нужно сложить, каждое из них не превышает 10000.

Формат выходных данных

В выходной файл выведите, сколько денег нам потребуется на нахождение суммы этих N чисел. Результат должен быть выведен с двумя знаками после десятичной точки.

Примеры

calc.in	calc.out
4 10 11 12 13	4.60
2 1 1	0.10

Задача C. Огромный граф

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

В этой задаче вам нужно найти кратчайший путь между двумя вершинами в огромном неориентированном невзвешенном графе. Граф настолько огромный, что мы даже не знаем, сколько в нем вершин.

Формат входных данных

В первой строке входных данных содержится одно целое число — количество ребер в графе M ($0 \leqslant M \leqslant 10^5$). Во второй строке записано два целых числа — номер начальной и номер конечной вершины. В следующих M строках заданы ребра графа — номера двух вершин, соединенных ребром. Все номера вершин — целые числа от 1 до 10^9.

Формат выходных данных

Если пути между данными вершинами нет, программа должна вывести число -1. Если путь существует, то программа должна вывести число l — длину кратчайшего пути между данными вершинами. В следующей строке программа должна вывести $l + 1$ число — номера вершин этого пути.

Примеры

hugegraph.in	hugegraph.out
5 1 4 1 3 3 2 2 4 2 1 2 3	2 1 2 4
4 20 30 20 10 20 40 40 30 10 30	2 20 10 30
0 1 1000000000	-1

Задача D. Рейсы во времени

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

Между N населёнными пунктами совершаются пассажирские рейсы на машинах времени.

В момент времени 0 вы находитесь в пункте A. Вам дано расписание рейсов. Требуется оказаться в пункте B как можно раньше (то есть в наименьший возможный момент времени).

При этом разрешается делать пересадки с одного рейса на другой. Если вы прибываете в некоторый пункт в момент времени T, то вы можете уехать из него любым рейсом, который отправляется из этого пункта в момент времени T или позднее (но не раньше).

Формат входных данных

Первая строка входного файла содержит число N — количество населённых пунктов ($1 \leqslant N \leqslant 1\,000$). Вторая строка содержит два числа A и B — номера начального и конечного пунктов. Третья строка содержит число K — количество рейсов ($0 \leqslant K \leqslant 1000$). Следующие K строк содержат описания рейсов, по одному на строке. Каждое описание представляет собой четвёрку целых чисел. Первое число каждой четвёрки задаёт номер пункта отправления, второе — время отправления, третье — пункт назначения, четвёртое — время прибытия. Номера пунктов — натуральные числа из диапазона от 1 до N. Пункт назначения и пункт отправления могут совпадать. Время измеряется в некоторых абсолютных единицах и задаётся целым числом, по модулю не превышающим 10^9. Поскольку рейсы совершаются на машинах времени, то время прибытия может быть как больше времени отправления, так и меньше, или равным ему.

Гарантируется, что входные данные таковы, что добраться из пункта A в пункт B всегда можно.

Формат выходных данных

Выведите в выходной файл минимальное время, когда вы сможете оказаться в пункте B.

Примеры

time.in	time.out
2 1 1 2 1 1 2 10 1 10 1 9	0
1 1 1 3 1 3 1 -5 1 -5 1 -3 1 -4 1 -10	-10

Задача E. Кратчайшие пути

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

Вам дан взвешенный ориентированный граф и вершина s в нём. Для каждой вершины графа u выведите длину кратчайшего пути от вершины s до вершины u.

Формат входных данных

Первая строка входного файла содержит три целых числа n, m, s — количество вершин и ребёр в графе и номер начальной вершины соответственно ($2 \leqslant n \leqslant 2\,000$, $1 \leqslant m \leqslant 5\,000$).

Следующие m строчек описывают рёбра графа. Каждое ребро задаётся тремя числами — начальной вершиной, конечной вершиной и весом ребра соответственно. Вес ребра — целое число, не превосходящее 10^{15} по абсолютной величине. В графе могут быть кратные рёбра и петли.

Формат выходных данных

Выведите n строчек — для каждой вершины u выведите длину кратчайшего пути из s в u. Если не существует пути между s и u, выведите «*». Если не существует кратчайшего пути между s и u, выведите «-».

Примеры

path.in	path.out
6 7 1 1 2 10 2 3 5 1 3 100 3 5 7 5 4 10 4 3 -18 6 1 -1	0 10 - - - *