Задача A. Дейкстра

Имя входного файла:	`dijkstra.in`
Имя выходного файла:	`dijkstra.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	64 мегабайта

Дан ориентированный взвешенный граф.

Найдите кратчайшее расстояние от одной заданной вершины до другой.

Формат входных данных

В первой строке входного файла три числа: N, S и F ($1 \leqslant N \leqslant 1000, 1 \leqslant S, F \leqslant N$), где N — количество вершин графа, S — начальная вершина, а F — конечная. В следующих N строках по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое целое неотрицательное число, не превосходящее 10 000 — присутствие ребра данного веса. На главной диагонали матрицы всегда нули.

Формат выходных данных

Вывести искомое расстояние или -1, если пути не существует.

Пример

dijkstra.in	dijkstra.out
3 1 2 0 -1 2 3 0 -1 -1 4 0	6

Задача B. Кратчайший путь

Имя входного файла:	`distance.in`
Имя выходного файла:	`distance.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

Дан неориентированный взвешенный граф.

Найти кратчайший путь между двумя данными вершинами.

Формат входных данных

Первая строка входного файла содержит натуральные числа N и M ($N \leqslant 2\,000$, $M \leqslant 50\,000$) — количество вершин и ребер графа. Вторая строка входного файла содержит натуральные числа S и F ($1 \leqslant S, F \leqslant N$, $S \neq F$) — номера вершин, длину пути между которыми требуется найти Следующие M строк по три натуральных числа b_i, e_i и w_i — номера концов i-ого ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant w_i \leqslant 100\,000$).

Формат выходных данных

Первая строка должна содержать одно натуральное число — длина минимального пути между вершинами S и F. Во второй строке через пробел выведите вершины на кратчайшем пути из S в F в порядке обхода. Если путь из S в F не существует, выведите -1.

Пример

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

Задача C. Флойд

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

Полный ориентированный взвешенный граф задан матрицей смежности. Постройте матрицу кратчайших путей между его вершинами.

Гарантируется, что в графе нет циклов отрицательного веса.

Формат входных данных

В первой строке вводится единственное число N ($1 \leqslant N \leqslant 100$) — количество вершин графа. В следующих N строках по N чисел задается матрица смежности графа (j-ое число в i-ой строке — вес ребра из вершины i в вершину j). Все числа по модулю не превышают 100. На главной диагонали матрицы — всегда нули.

Формат выходных данных

Выведите N строк по N чисел — матрицу расстояний между парами вершин, где j-ое число в i-ой строке равно весу кратчайшего пути из вершины i в j.

Пример

floyd.in	floyd.out
4 0 5 9 100 100 0 2 8 100 100 0 7 4 100 100 0	0 5 7 13 12 0 2 8 11 16 0 7 4 9 11 0

Задача D. Цикл отрицательного веса

Имя входного файла:	negcycle.in
Имя выходного файла:	negcycle.out
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

Дан ориентированный граф. Определите, есть ли в нем цикл отрицательного веса, и если да, то выведите его.

Формат входных данных

Во входном файле в первой строке число N ($1 \leqslant N \leqslant 80$) — количество вершин графа. В следующих N строках находится по N чисел — матрица смежности графа. Все веса ребер не превышают по модулю 10 000. Если ребра нет, то соответствующее число равно 100 000.

Формат выходных данных

В первой строке выходного файла выведите «YES», если цикл существует или «NO» в противном случае. При его наличии выведите во второй строке количество вершин в искомом цикле и в третьей строке — вершины, входящие в этот цикл в порядке обхода.

Пример

negcycle.in	negcycle.out
2 0 -1 -1 0	YES 2 2 1

Задача E. Заправки

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

В стране N городов, некоторые из которых соединены между собой дорогами. Для того, чтобы проехать по одной дороге, требуется один бак бензина. В каждом городе бак бензина имеет разную стоимость. Вам требуется добраться из первого города в N-й, потратив как можно меньшее количество денег.

Дополнительно имеется канистра для бензина, куда входит столько же бензина, сколько входит в бак. В каждом городе можно заправить бак, залить бензин в канистру, залить и туда и туда, или же перелить бензин из канистры в бак.

Формат входных данных

В первой строке входного файла записано N чисел ($1 \leqslant N \leqslant 25$), i-ое из которых задает стоимость бензина в i-ом городе (все числа целые в диапазоне от 0 до 100).

Во следующих строках описаны все дороги (по одной в строке). Каждая дорога задается двумя числами — номерами городов, которые она соединяет. Все дороги двухсторонние, между двумя городами существует не более одной дороги, не существует дорог, ведущих из города в себя.

Формат выходных данных

В выходной файл выведите одно число — суммарную стоимость маршрута или -1, если добраться до нужного города невозможно.

Пример

gasstation.in	gasstation.out
1 10 2 15 1 2 1 3 4 2 4 3	2