TopSort1, TopSort2. Топологическая сортировка 1, 2

Имя входного файла: `topsort1.in, topsort2.in`
Имя выходного файла: `topsort1.out, topsort2.out`

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входного файла

В первой строке входного файла даны два натуральных числа N и M — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Варианты

topsort1: $1 \leqslant N \leqslant 200$, $0 \leqslant M \leqslant N^2$.
topsort2: $1 \leqslant N \leqslant 100\,000$, $0 \leqslant M \leqslant 100\,000$.

Формат выходного файла

Вывести любую топологическую сортировку графа в виде последовательности номеров вершин. Если граф невозможно топологически отсортировать, вывести -1.

Пример

topsort1.in, topsort2.in	topsort1.out, topsort2.out
6 6 1 2 3 2 4 2 2 5 6 5 4 6	4 6 3 1 2 5
3 3 1 2 2 3 3 1	-1

Console1. Поиск набора образцов 1

Имя входного файла: `console1.in`
Имя выходного файла: `console1.out`

Напишите программу, которая для каждой строки из заданного набора S проверяет, верно ли, что она содержит как подстроку одну из строк из набора T.

Формат входного файла

Первая строка входного файла содержит натуральное число n ($n \leqslant 100$) — количество строк в наборе T. Каждая из следующих n строк содержит непустую строку длины не более 80-ти символов.

Оставшаяся часть файла содержит строки из набора S. Каждая строка состоит из ASCII символов с кодами от 32 до 126 включительно. Строка может быть пустой; ганатируется, что длины строк не превышают 250-ти символов.

Гарантируется, что размер входного файла не превышает 1 Мбайт.

Формат выходного файла

В выходной файл выведите все строки из набора S (в том порядке, в котором они находятся во входном файле), содержащие как подстроку по крайней мере одну строку из набора T.

Пример

console1.in	console1.out
3 gr sud abc lksh sudislavl kostroma summer group b	sudislavl group b

Mincoin1, mincoin2. Сдачи нет! 1, 2

Имя входного файла: `mincoin1.in, mincoin2.in`
Имя выходного файла: `mincoin1.out, mincoin2.out`

Саша опаздывает на автобусы, уходящие в ЛКШ. К сожалению, ему недостаточно просто успеть на автобус — по дороге ему еще надо купить подарок девушке Насте, у которой сегодня день рождения. По дороге ему встретился магазин, где он может купить подарок. Подарок стоит c рублей. У Саши в кошельке есть n купюр, но у продавца нет сдачи, и поэтому Саша должен набрать нужную сумму без сдачи. Более того, у Саши очень мало времени, деньги надо достать как можно быстрее, и поэтому Саша хочет дать продавцу нужную сумму минимальным количеством купюр. Помогите ему сделать это.

Формат входного файла

Первая строка входного файла содержит три целых неотрицательных числа n, c и k ($0 \leqslant n \leqslant 1000$, $0 \leqslant c \leqslant 1000$, $1 \leqslant k \leqslant 1\,000\,000\,000$). Далее следуют n чисел a_1, a_2, …, a_n ($1 \leqslant a_i \leqslant 1000$) — номиналы купюр в том порядке, как они лежат у Саши в кошельке.

Формат выходного файла

В первую строку выходного файла выведите одно число m — минимальное количество купюр, которыми Саша может набрать необходимую сумму. Во вторую строку выведите m чисел — номера купюр, которые должен дать продавцу Саша. Саше надо будет доставать купюры последовательно из кошелька, поэтому номера должны быть выведены в возрастающем порядке.

Если набрать необходимую сумму без сдачи невозможно, то выведите в выходной файл одно число -1.

Варианты

mincoin1: Первому Саше безразлично, как набирать требуемую сумму. Поэтому, если в задаче mincoin1 есть несколько решений, вы можете вывести произвольное. Ваша программа может не обращать внимания на значение числа k.

mincoin2: Но другому Саше не все равно, как набирать необходимою сумму. У другого Саши есть любимое число k, и поэтому, если в задаче mincoin2 решение существует, вы должны вывести k-ое в лексикографическом порядке. Лексикографический порядок в этой задаче мы будем понимать как обычно для последовательностей из m чисел, а именно: все различные решения сортируем по номеру первой входящей в решение купюры, при равных номерах первой купюры — по второй и т.д. В этом варианте гарантируется, что, если решение существует, то k не превосходит общего количества решений. Решения нумеруются, начиная с 1; решения, отличающиеся порядком купюр, считаем за одно решение (в выходной файл вы должны в любом случае выводить номера купюр в возрастающем порядке). Обратите также внимание, что, если решения не существует, то никаких дополнительных ограничений на значение k во входном файле не налагается.

Пример

mincoin1.in, mincoin2.in	mincoin1.out, mincoin2.out
5 5 2 1 4 2 3 1	2 2 5
5 100 2 1 4 2 3 1	-1
2 1 100 2 3	-1

Примечание: в первом примере все возможные варианты решения, отсортированные в лексикографическом порядке, следующие:

1 2 (первая и вторая купюры, т.е. купюры номиналом 1 и 4)

2 5 (вторая и пятая купюры, т.е. купюры номиналом 4 и 1)

3 4 (третья и четвертая купюры, т.е. купюры номиналом 2 и 3)

В варианте mincoin1 вы можете вывести любой из этих вариантов, в варианте mincoin2 — только второй.

Msquare. Магический квадрат

Имя входного файла: `msquare.in`
Имя выходного файла: `msquare.out`

Магическим квадратом будем называть квадрат с одинаковой суммой чисел по всем вертикалям и горизонталям; никаких требований на суммы по диагоналям накладывать не будем. Составьте такой квадрат из заданного набора чисел.

Формат входного файла

Во входном файле записаны 16 различных целых чисел в интервале от 0 до 32 768.

Формат выходного файла

В выходной файл необходимо вывести искомое расположение чисел, составляющее магический квадрат 4×4 (каждое число должно встречаться ровно один раз), в четырех строках по четыре числа, или строку '`NO SOLUTION`', если квадрат составить нельзя.

Пример

msquare.in	msquare.out
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	1 2 15 16 6 11 7 10 13 12 4 5 14 9 8 3