Задача A. Число путей в ациклическом графе (1 балл)

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

Задан ориентированный ациклический связный граф. Найдите количество различных путей из вершины с номером 1 в вершину с номером n.

Формат входного файла

В первой строке входных данных содержится два целых числа n и m — количество вершин и ребер графа, ($2 \leq n \leq 10^5$, $2 \leq m \leq 2 \cdot 10^5$). В следующих m строках содержится по два целых числа: номера вершин, которые соединяет соответствующее ребро графа.

Формат выходного файла

В первой строке выведите количество различных путей из вершины с номером 1 в вершину с номером n. Ответ следует выводить по модулю $10^9 + 7$.

Пример

countpaths.in	countpaths.out
4 4 1 2 1 3 3 2 2 4	2

Задача B. Наибольшая возрастающая подпоследовательность (1 балл)

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

Дана последовательность, требуется найти её наибольшую возрастающую подпоследовательность.

Формат входного файла

В первой строке входных данных задано целое число N — длина последовательности ($1 \leq N \leq 5000$). Во второй строке задается сама последовательность. Числа разделяются пробелом. Элементы последовательности — целые числа, не превосходящие 10^9 по модулю.

Формат выходного файла

В первой строке выведите длину наибольшей возрастающей подпоследовательности, а во второй строке выведите через пробел саму наибольшую возрастающую подпоследовательность данной последовательности. Если ответов несколько — выведите любой.

Пример

lis.in	lis.out
6 3 29 5 5 28 6	3 3 5 28

Задача C. Наибольшая общая подпоследовательность (1 балл)

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

Даны две последовательности, требуется найти и вывести их наибольшую общую подпоследовательность.

Формат входного файла

В первой строке входных данных содержится целое число N — длина первой последовательности ($1 \leq N \leq 2000$). Во второй строке заданы члены первой последовательности (через пробел) — целые числа, не превосходящие 10^9 по модулю. В третьей строке записано целое число M — длина второй последовательности ($1 \leq M \leq 2000$). В четвертой строке задаются члены второй последовательности (через пробел) — целые числа, не превосходящие 10^9 по модулю.

Формат выходного файла

В первой строке выведите длину наибольшей общей подпоследовательности, а во второй строке выведите через пробел саму наибольшую общую подпоследовательность данных последовательностей. Если ответов несколько — выведите любой.

Пример

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

Задача D. Рюкзак (1 балл)

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

Дано n предметов массой $m_1, \ldots, m_n$ и стоимостью $c_1, \ldots, c_n$ соответственно.

Ими наполняют рюкзак, который выдерживает вес не более m. Определите набор предметов, который можно унести в рюкзаке, имеющий наибольшую стоимость.

Формат входного файла

В первой строке вводится натуральное число n, не превышающее 1000 и натуральное число m, не превышающее 10000.

Во второй строке вводятся n натуральных чисел m_i, не превышающих 100.

Во третьей строке вводятся n натуральных чисел $_i$, не превышающих 100.

Формат выходного файла

В первой строке выведите количество предметов, которые нужно взять. Во второй строке выведите номера предметов (числа от 1 до n), которые войдут в рюкзак наибольшей стоимости.

Пример

knapsack.in	knapsack.out
4 6 2 4 1 2 7 2 5 1	3 1 3 4

Задача E. Расстояние Левенштейна (1 балл)

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

Дана текстовая строка. С ней можно выполнять следующие операции:

- Заменить один символ строки на другой символ.
- Удалить один произвольный символ.
- Вставить произвольный символ в произвольное место строки.

Например, при помощи первой операции из строки «СОК» можно получить строку «СУК», при помощи второй операции — строку «ОК», при помощи третьей операции — строку «СТОК». Минимальное количество таких операций, при помощи которых можно из одной строки получить другую, называется стоимостью редактирования или расстоянием Левенштейна. Определите расстояние Левенштейна для двух данных строк.

Формат входного файла

Программа получает на вход две строки, длина каждой из которых не превосходит 5000 символов, строки состоят только из заглавных латинских букв.

Формат выходного файла

Требуется вывести одно число — расстояние Левенштейна для данных строк.

Пример

levenshtein.in	levenshtein.out
ABCDEFGH ACDEXGIH	3

Задача F. Умножение матриц (1 балл)

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

В произведении последовательности матриц полностью расставлены скобки, если выполняется один из следующих пунктов:

- Произведение состоит из одной матрицы.
- Оно является заключенным в скобки произведением двух произведений с полностью расставленными скобками.

Полная расстановка скобок называется оптимальной, если количество операций, требуемых для вычисления произведения, минимально.

Требуется найти оптимальную расстановку скобок в произведении последовательности матриц.

Формат входного файла

В первой строке входных данных содержится целое число n — количество матриц ($1 \le n \le 400$). В n следующих строк содержится по два целых числа a_i и b_i — количество строк и столбцов в i-ой матрице соответственно ($1 \le a_i, b_i \le 100$). Гарантируется, что $b_i = a_{i+1}$ для любого $1 \le i \le n - 1$

Формат выходного файла

В выходной файл выведите оптимальную расстановку скобок. Если таких расстановок несколько, выведите любую.

Пример

matrix.in	matrix.out
3 10 50 50 90 90 20	((AA)A)

Примечание

В данном примере возможно две расстановки скобок: $((AA)A)$ и $(A(AA))$. При первой количество операций будет равно $10 \cdot 50 \cdot 90 + 10 \cdot 90 \cdot 20 = 63000$, а при второй — $10 \cdot 50 \cdot 20 + 50 \cdot 90 \cdot 20 = 100000$.

Задача G. Максимальный подпалиндром (1 балл)

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

Палиндромом называется строка, которая одинаково читается как слева направо, так и справа налево. Подпалиндромом данной строки называется последовательность символов из данной строки, не обязательно идущих подряд, являющаяся палиндромом. Например, «HELOLEH» является подпалиндромом строки «HTEOLFEOLEH». Напишите программу, находящую в данной строке подпалиндром максимальной длины.

Формат входного файла

Во входном файле находится строка длиной не более 2000 символов, состоящая из заглавных букв латинского алфавита.

Формат выходного файла

Выведите на первой строке выходного файла длину максимального подпалиндрома, а на второй строке сам максимальный подпалиндром. Если таких подпалиндромов несколько, то ваша программа должна вывести любой из них.

Пример

palindrome.in	palindrome.out
HTEOLFEOLEH	7 HEOLOEH

Задача H. Паросочетание максимального веса в дереве (2 балла)

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

Паросочетанием называется такое подмножество рёбер графа, в котором у любых двух рёбер нет общей вершины. Дан взвешенный неориентированный граф, в котором между каждой парой вершин существует ровно один простой путь. Требуется в заданном графе найти паросочетание максимального веса.

Формат входного файла

В первой строке входных данных содержится число n — количество вершин в графе ($2 \le n \le 10^5$). Следующие $n-1$ строк содержат описание ребер. Каждое ребро задаётся стартовой вершиной, конечной вершиной и весом ребра w_i ($1 \le w_i \le 10^9$).

Формат выходного файла

Требуется вывести одно число — вес максимального паросочетания.

Пример

matching.in	matching.out
4 1 2 10 1 3 1 3 4 1	11

Задача I. Задача коммивояжера (2 балла)

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

Вам дан неориентированный взвешенный граф без петель и кратных ребер. Необходимо найти в нем путь наименьшего веса, который проходит по всем вершинам ровно один раз.

Формат входного файла

В первой строке находятся два целых числа n и m — количество вершин и ребер в графе ($1 \le n \le 18$, $0 \le m \le \frac{n \cdot (n-1)}{2}$). Следующие m строк содержат описания ребер: три целых числа a_i, b_i, w_i, обозначающих соответственно пару вершин и вес ребра, соединяющего эти вершины ($1 \le a_i, b_i \le n$, $1 \le w_i \le 10^8$).

Формат выходного файла

Выведите в выходной файл одно число — вес искомого пути. Если такого пути не существует, выведите -1.

Примеры

salesman.in	salesman.out
4 6 1 2 20 1 3 42 1 4 35 2 3 30 2 4 34 3 4 12	62
4 3 1 2 1 1 3 1 1 4 1	-1

Задача J. Оптимальный префиксный код с сохранением порядка (2 балла)

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

Дан алфавит Σ, где каждому символу c_i сопоставим его код p_i. Кодирование называется оптимальным префиксным с сохранением порядка, если соблюдаются:

1. $\forall i, j : c_i < c_j \iff p_i < p_j$.
2. $\nexists i, j : i \neq j, p_i$ — префикс p_j.
3. $\sum_{i=1}^{|\Sigma|} f_i \cdot |p_i|$ — минимально, где f_i — частота встречаемости символа c_i в тексте, а $|p_i|$ — длина его кода.

В алфавите n символов. i-й в лексикографическом порядке символ встречается в тексте f_i раз. Постройте оптимальный префиксный код с сохранением порядка.

Формат входного файла

В первой строке дано целое число n ($1 \leq n \leq 2000$) — количество символов в алфавите. Далее в n строках записаны целые числа f_i ($1 \leq f_i \leq 10^6$) обозначающие количество раз, которое встречается соответствующий символ в тексте.

Формат выходного файла

В первой строке выведите значение $\sum_{i=1}^{|\Sigma|} f_i \cdot |p_i|$. В $i+1$-й строке выведите код, который соответствует i-у символу. Если существует несколько решений, выведите любое.

Примеры

optimalcode.in	optimalcode.out
3 10000 1 1	10004 0 10 11
5 15 1 2 3 4	44 0 1000 1001 101 11