

Задача А. Редакционное расстояние

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

Рассмотрим две строки S и T , состоящие из символов с ASCII кодами от 33 до 126 включительно. Редакционное расстояние между строками — минимальное количество вставок, удалений и замен символов, которые необходимы для того, чтобы преобразовать одну из этих строк в другую. Вам необходимо найти редакционное расстояние между S и T или сообщить, что оно больше заданного числа k .

Формат входного файла

В первой строке входного файла задана строка S , во второй — T ($0 \leq |S|, |T| \leq 100\,000$), в третьей — целое число k ($0 \leq k \leq 50$).

Формат выходного файла

Если искомое редакционное расстояние больше k , то выведите в первую строку “Infinity”, в противном случае выведите редакционное расстояние.

Программы, всегда выводящие “Infinity”, будут оценены в 0 баллов.

Примеры

distance.in	distance.out
abacabadabacaba abacabadabacaba 24	0
abcdef acdefu 7	2
aaaaaaaaa bbbbbbbbbb 5	Infinity

Задача В. Какуро

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

Какуро — головоломка с числами, которую можно назвать математическим аналогом кроссворда. Название Какуро происходит от сокращения японского назва-

ния kasan kurosu (перекрёстное сложение); в США головоломка также известна под названием “Cross Sums” («Пересекающиеся суммы»).

Поле состоит из клеток чёрного и белого цвета. Несколько белых клеток, идущих подряд по горизонтали или по вертикали, называются *блоком*. Про каждый блок известна сумма цифр, которые должны стоять в этом блоке.

Во все белые клетки нужно вписать по одной цифре от 1 до 9 так, чтобы, во-первых, сумма цифр в каждом блоке сошлась с указанным числом, а во-вторых, чтобы в каждом блоке все цифры были различны.

Формат входного файла

В первой строке входного файла заданы два целых числа W и H — количество столбцов и строк, соответственно. Во второй строке находится целое число N — количество блоков в головоломке.

В каждой из следующих N строк находятся описания блоков, по одному на строку. Каждое описание состоит из пяти целых чисел. Первые четыре из них задают нижнюю-левую и верхнюю-правую клетки блока, пятое — сумму цифр в этом блоке. Нумерация ведётся от 1 до W по горизонтали и от 1 до H по вертикали. $2 \leq W, H \leq 11$. $1 \leq N \leq 25$.

Одна из размерностей каждого блока равна 1, другая всегда больше 1. Блоки одинаковой ориентации не могут иметь общих клеток.

Формат выходного файла

Если решение существует, выведите H строк по W чисел. Если на соответствующей клетке в решении стоит одна из цифр от 1 до 9, следует вывести её. Иначе на этой позиции следует вывести 0. Если решение неоднозначно, разрешается выводить любое.

Если решения не существует, следует вывести “No solution” (без кавычек).

Программы, всегда выводящие “No solution”, будут оценены в 0 баллов.

Примеры

kakuro.in	kakuro.out
5 5 2 2 2 2 3 17 2 2 3 2 16	0 0 0 0 0 0 9 7 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0
5 5 2 2 2 2 3 17 2 2 3 2 18	No solution

Задача С. Разбиения на пары

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

Разбием на пары множества чисел $\{1, 2, \dots, 2n\}$ назовём набор из n пар чисел, в котором каждое число от 1 до $2n$, включительно, встречается ровно один раз. Записывается разбиение на пары так: сначала числа внутри каждой пары сортируются по возрастанию, затем сами пары сортируются по возрастанию первых чисел, и наконец, пары выписываются в строчку по порядку. Разбиения считаются одинаковыми, если они имеют одинаковую запись.

Так, например, для $n = 2$ существует всего три различных разбиения множества чисел $\{1, 2, 3, 4\}$ на пары. Они записываются так:

(1 2) (3 4)
(1 3) (2 4)
(1 4) (2 3)

Вася и Петя поспорили, кто из них быстрее выпишет все разбиения на пары множества чисел $\{1, 2, \dots, 2n\}$. Оба одновременно берут по листку бумаги и начинают выписывать все разбиения на пары в лексикографическом порядке. Одно разбиение идёт в этом порядке раньше другого, если для какого-то k от 0 до $2n - 1$, включительно, первые k чисел в их записи совпадают, а $(k + 1)$ -е число первого разбиения меньше, чем $(k + 1)$ -е число второго.

Оказалось, что Вася выписывает одно разбиение ровно две секунды, а Петя — ровно одну секунду. Вася хочет знать, какое разбиение B Петя выписывал в ту секунду, когда сам Вася заканчивал выписывать некоторое разбиение A . Напишите программу, которая выяснит это для него, чтобы не отвлекать Васю от процесса выписывания.

Формат входного файла

В первой строке входного файла задано целое число n ($1 \leq n \leq 17$). Во второй строке записаны $2n$ целых чисел через пробел — разбиение A , выписанное Васей. Скобки в записи разбиения опущены для удобства чтения.

Формат выходного файла

Если Петя уже выписал все разбиения до той секунды, в которую Вася закончил выписывать разбиение A , выведите в первой строке выходного файла фразу “Already finished!” без кавычек. В противном случае выведите в первой строке

выходного файла $2n$ целых чисел через пробел — запись разбиения B , выписанного Петей, также без скобок.

Программы, всегда выводящие “Already finished!”, будут оценены в 0 баллов.

Примеры

pairings.in	pairings.out
2 1 2 3 4	1 3 2 4
2 1 3 2 4	Already finished!