Unfold. Распаковка строки

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

Одним из простейших способов сжатия строк является сжатие повторяющихся подстрок. Рассмотрим один из простейших таких способов сжатия. Исходная строка для такого способа состоит только из заглавных латинских букв, результат сжатия состоит из заглавных латинских букв, цифр и круглых скобок. Обозначим результат распаковки сжатой строки S как $u(S)$, тогда функция u определяется следующим образом:

- Если S состоит только из заглавных латинских букв (в том числе если S — пустая строка), то $u(S) = S$;
- Если S имеет вид $X(Q)$, где X — целое число, а Q — корректная сжатая строка, то $u(S)$ — это строка $u(Q)$, повторенная X раз:

$$u(S) = \underbrace{u(Q)u(Q)\dots u(Q)}_{X \text{ раз}}$$

- Если S имеет вид PQ, где P и Q — две непустые корректные сжатые строки, то $u(S)$ — это строка $u(P)$, к которой приписана строка $u(Q)$:

$$u(S) = u(P)u(Q)$$

- Если строку S невозможно представить в одном из перечисленных выше видов, то S не является корректной сжатой строкой.

Например, строка `10(A)2(BA)B2(C)D` распаковывается в `AAAAAAAAAABABABCCD`.

Напишите программу, которая будет выполнять распаковку данной строки.

Формат входного файла

Входной файл содержит одну строку S. Строка состоит только из латинских букв, цифр и круглых скобок, и имеет длину не менее 1 и не более 100 символов.

Формат выходного файла

Выведите в выходной файл одно слово — результат распаковки строки S. Если строка S не является корректной сжатой строкой, то выведите в выходной файл одну строку '`BOTVA`'.

Гарантируется, что длина результата распаковки строки не превосходит 100 символов.

Пример

unfold.in	unfold.out
9(A)3(AB)CCD	AAAAAAAAAABABABCCD
2(NEERC3(YES))	NEERCYESYESYESNEERCYESYESYES
2(ZZ	BOTVA
(A)	BOTVA
2(3()W)	WW

Distance. Расстояние между многоугольниками

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

Нужно найти расстояние между двумя выпуклыми непересекающимися многоугольниками.

Формат входного файла

Во входном файле содержатся описания двух многоугольников.

Многоугольник задается числом вершин N ($1 \leqslant N \leqslant 50\,000$) и координатами N вершин. Вершины даны в порядке обхода по часовой стрелке. Координаты целые и не превосходят 10^9 по модулю.

В обоих многоугольниках никакие три точки не лежат на одной прямой.

Формат выходного файла

Выведите одно вещественное число — расстояние между многоугольниками. Выводите ответ с максимально возможной точностью. Ваше решение будет считаться верным, если относительная или абсолютная погрешность ответа не превосходит 10^{-12}.

Пример

distance.in	distance.out
4 0 0 0 1 1 1 1 0 3 2 0 2 2 4 0	1.0000000000000000000

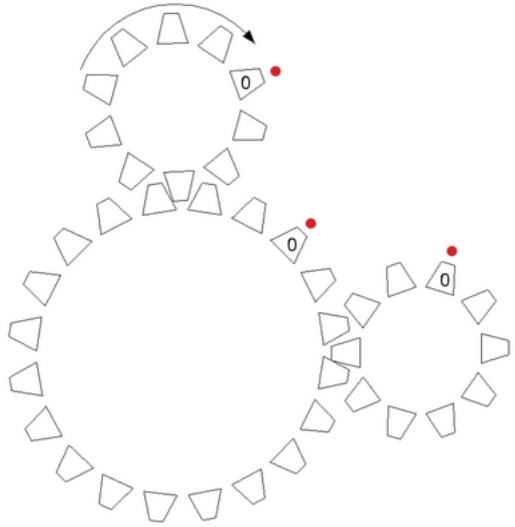
Gears. Шестеренки

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

В Словацком Институте Механизмов изобрели специальное устройство, единственная задача которого — вращать много шестеренок. После того, как изобретатель Петер в течение трех часов наблюдал за работой этого устройства, он заинтересовался одной математической проблемой.

В устройстве N шестеренок, соединенных последовательно друг с другом — i-я шестеренка имеет a_i зубчиков, пронумерованных против часовой стрелки числами от 0 до $a_i - 1$. Она соединяется с $(i-1)$-й и $(i+1)$-й (исключениями являются первая и последняя шестеренки, которые соединяются только с одной соседней). Рядом с каждой шестеренкой нарисована красная отметка. В начале работы все шестеренки повернуты так, что 0-й зубчик находится рядом с соответсвующей красной отметкой.

Скрытый механизм вращает первую шестеренку по часовой стрелке со скоростью один зубчик в секунду. Поскольку все шестеренки соединены, то они вращаются с такой же скоростью.



Петер придумал конфигурацию шестеренок, которую он хочет получить. В его конфигурации для каждого i известен номер зубчика b_i, который должен располагаться рядом с красной отметкой, соответсвующей i-й шестеренке. Теперь его интересует получится ли когда-нибудь придуманная конфигурация и, если да, то когда это случится в первый раз.

Формат входного файла

В первой строке содержится число тестов M ($1 \leqslant M \leqslant 100$). Для каждого теста в отдельной строке задается число шестеренок N ($1 \leqslant N \leqslant 500$). Следующие N строк содержат описания шестеренок a_i и b_i ($1 \leqslant a_i \leqslant 500\,000\,000, 0 \leqslant b_i < a_i$).

Формат выходного файла

Для каждого теста выведите минимальное время когда будет получена требуемая конфигурация. Если этого невозможно, то выведите `Impossible`.

Гарантируется, что ответ и необходимые промежуточные результаты помещаются в 64-х битные целые числа.

Примеры

gears.in	gears.out
2 3 5 2 4 2 6 4 2 4 3 6 0 3 111103 15028 111103 96075 1217 864 2 51047849 10084861 51047849 40962988	22 Impossible 19902465 10084861

Divisible. ДКА, проверяющий делимость

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

В условиях данной задачи разрешим в двоичной записи чисел присутствие ведущих нулей. Кроме того, разрешим записывать число 0 в двоичной записи как ϵ (то есть как пустую строку).

Таким образом, строки «101» и «000101» являются двоичными записями числа 5, а строки «000000» и «» — двоичными записями числа 0.

Дано множество положительных нечетных чисел S.

Постройте ДКА над алфавитом $\{0, 1\}$, принимающий те и только те строки, которые являются двоичными записями чисел, делящихся на хотя бы одно число из множества S.

Число состояний в построенном автомате не обязано быть минимальным, но не должно превышать 20 000. Гарантируется, что существует искомый ДКА с не более чем 1 000 состояний.

Формат входного файла

В первой строке входного файла содержится число $|S|$ ($1 \leqslant |S| \leqslant 1\,000$) — мощность

множества S. Затем следуют $|S|$ различных положительных нечетных чисел, не превышающих 10^9.

Формат выходного файла

Первая строка выходного файла должна содержать число $|Q|$ — количество состояний автомата. Состояния нумеруются числами от 1 до $|Q|$.

Следующая строка должна содержать число q_0 ($1 \leqslant q_0 \leqslant |Q|$) — номер начального состояния, затем число $|T|$ — количество терминальных состояний, затем $|T|$ чисел от 1 до $|Q|$ — номера терминальных состояний.

Следующие $|Q|$ строк должны содержать по $|\delta|$ чисел — описание функции переходов δ. (Для каждого состояния в отдельной строке приводятся номера состояний, в которые из него ведут переходы по всем символам алфавита).

Пример

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

Guard. Кладбищенский сторож

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

Где отбой в двадцать три наступает всегда...

В одном чёрном-чёрном лесу, на чёрном-чёрном кладбище стояло надгробие из чистого золота. Это надгробие охраняли две страшных, чёрных-пречёрных собаки. Каждая собака сидит на цепи около чёрного-чёрного столба, а неподалеку, в том же лесу, стоит еще и чёрный-чёрный домик сторожа. Каждое утро сторож выходит из домика и несёт собакам миски с едой. Он ставит миски на землю так, чтобы собаки могли есть, оставаясь на привязи у своих столбов. Вычислите кратчайшее расстояние, которое необходимо пройти сторожу, чтобы накормить обеих собак и вернуться в домик. Сторож легко может нести еду сразу для двух собак и может кормить их в любом порядке.

Формат входного файла

В первой строке записаны три целых числа: расстояние в метрах от домика сторожа до первого столба R_1, от домика сторожа до второго столба R_2, и расстояние между столбами R_3. Во второй строке одно целое число — длина цепи каждой из собак в метрах R_4 (цепи у собак одинаковые). Все числа R_i ($i = 1, 2, 3$) удовлетворяют ограничению $0 \leqslant R_i \leqslant 10000; 1 \leqslant R_4 \leqslant 10000$.

Формат выходного файла

Выведите одно число — длину кратчайшего пути сторожа в метрах с точностью 3 знака после запятой.

Пример

guard.in	guard.out
1000 2000 1000 250	3500.000