Задача A. Ладейная постановка

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Рома хочет снять видео, на котором шахматные ладьи красиво передвигаются на доске $w \times h$ под музыку. Его план таков: под каждый такт музыки либо на доске появляется новая ладья, либо уже стоящая на доске ладья передвигается по горизонтали или по вертикали на новую позицию. При этом можно перескакивать через другие ладьи, но нельзя передвигаться на клетку, на которой стоит или ранее хоть раз стояла ладья.

Рома ввёл меру визуальной утонченности для каждой клетки доски. И хочет, чтобы ладья передвигалась только в клетку с большей мерой утонченности, чем та, на которой она стояла до хода.

Какое минимальное число ладей достаточно для съемки такого видео, чтобы каждая клетка была посещена ладьей?

Формат входных данных

В первой строке заданы два целых числа h и w ($1 \leqslant h, w \leqslant 100$). Затем в h строчках идет описание матрицы визуальной утонченности. В каждой строке w целых чисел от 1 до $h \times w$. Все числа различные.

Формат выходных данных

В первой строке выходного файла выведите минимальное количество ладей k, которое необходимо поставить на поле.

В каждой из следующих k строк выведите описание пути для ладьи — значения утонченности клеток, посещаемых ладьей, в порядке обхода.

Если искомых ответов несколько, выведите любой из них.

Примеры

stdin	stdout
3 3 6 3 2 5 4 8 1 9 7	3 2 3 6 4 5 8 1 7 9

Задача B. Окружность и минимумы

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Найти минимум... найти максимум... Как это скучно. Давайте лучше искать отрезок окружности, на котором разность максимума и минимума максимальна.

Задача: нужно найти отрезок окружности, на котором сумма чисел не более x и при этом разность максимума и минимума максимальна. Гарантируется, что хотя бы один отрезок с суммой не более x существует.

Формат входных данных

Входные данные состоят из одного или нескольких тестов. Каждый тест описывается следующим образом: на первой строке количество чисел на окружности n ($1 \leqslant n \leqslant 300\,000$) и максимально допустимая сумма x ($0 \leqslant x \leqslant 10^{18}$). На второй строке числа на окружности в порядке обхода — $a_1, a_2, \ldots, a_n$ ($0 \leqslant a_i \leqslant 10^9$). Сумма всех n не превосходит 300 000.

Формат выходных данных

Выведите ответ на каждый из тестов. Ответ на тест должен выглядеть следующим образом: выведите на одной строке l и r (целые числа от 1 до n) — отрезок, на котором достигается максимум разности. Если оптимальных отрезков несколько, выведите любой. Если Вы выведите r меньше l, это будет означать, что отрезок состоит из элементов $a_l, a_{l+1}, \ldots, a_n, a_1, a_2, \ldots, a_r$, иначе отрезок состоит из элементов $a_l, a_{l+1}, \ldots, a_r$.

Примеры

stdin	stdout
6 4 4 1 2 1 3 1 6 4 1 4 1 2 1 3	4 5 5 6

Задача C. Такие разные пути

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вы когда-нибудь задумывались о том, что в неориентированном графе между двумя заданными вершинами может быть несколько разных путей? А вот. Бывает и такое. А раз бывает, обязательно нужно уметь такие пути искать.

Задача: дан неориентированный граф без петель и кратных ребер, две вершины s и t, число k от 1 до 3. Простым путем из s в t называется последовательность вершин $a_1, a_2, \ldots, a_n$, $a_1 = s$, $a_n = t$, при этом $\forall i \neq j$: $a_i \neq a_j$. Нужно найти и вывести k простых путей из s в t. Все k путей должны быть попарно различны. Пути $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_m$ называются различными, если $n \neq m$ или $\exists i$: $a_i \neq b_i$.

Формат входных данных

Входные данные состоят из одного или нескольких тестов. Каждый тест описывается следующим образом: на первой строке 5 целых чисел — количество вершин n, количество ребер m, начальная вершина s, конечная вершина t, количество путей k ($1 \leqslant s, t \leqslant n \leqslant 300\,000$, $0 \leqslant m \leqslant 300\,000$, $1 \leqslant k \leqslant 3$). Следующие m строк содержат пары чисел a и b от 1 до n — ребра графа. Гарантируется, что в графе нет ни петель, ни кратных ребер. Гарантируется, что s и t различны. Сумма n по всем тестам не превосходит 300 000. Сумма m по всем тестам не превосходит 300 000.

Формат выходных данных

Выведите ответ на каждый из тестов. Если существуют k различных простых путей из s в t, выведите на первой строке `YES`, иначе `NO`. Если ответ `YES`, выведите в следующих k строках найденные простые пути. Путь задается количеством вершин в нем и последовательностью вершин, включая концы.

Примеры

stdin	stdout
3 3 1 2 2 1 2 2 3 3 1 3 3 1 2 3 1 2 2 3 3 1	YES 2 1 2 3 1 3 2 NO

Задача D. Велорикши в Цзиньджене

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В городе Цзиньджень есть N велорикш, каждый из которых дежурит в определённой точке и знает наизусть некоторый маршрут. К сожалению, не зная местного языка, вы не можете просить велорикшу остановиться на полпути, так что вынуждены доезжать до конца его маршрута, если прибегнули к его услугам.

Кроме того, вы можете передвигаться по городу между любыми двумя точками — для упрощения модели будем считать, что время передвижения равняется эвклидовому расстоянию между точками начала и конца движения.

За какое наименьшее время можно добраться из точки A в точку B?

Формат входных данных

В первой строке содержатся целые числа x_A, y_A — координаты точки, из которой вы начинаете движение, во второй строке — целые числа x_B, y_B — координаты точки, в которую вы хотите добраться. В третьей строке целое число n ($1 \leqslant n \leqslant 100$) — количество велорикш. Далее n строк по пять целых чисел: $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2}$, t — координаты начала и конца маршрута велорикши и время проезда по маршруту ($0 < t \leqslant 1500$).

Все координаты принадлежат интервалу $[0, 1000]$.

Формат выходных данных

Выведите одно число — минимальное время перемещения из точки A в точку B. Допустимая погрешность ответа — 10^{-4}.

Примеры

stdin	stdout
0 0 5 5 1 0 1 5 1 1	6.0
0 0 5 5 1 1 1 15 14 1	7.0710678118654755

Задача E. Площадь неубитого слона

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Известный тайский охотник на слонов Тан Ке Ви планирует заполучить в свою коллекцию ещё один превосходный экземпляр шкуры слона. В предвкушении он даже высунулся из укрытия, и тут неожиданная мысль поразила его. «Что, если этот уникальный слон не влезет в мою гостиную?», — думал Тан, и с каждой секундой шансы совершить удачный выстрел таяли...

Помогите Тану, вычислите площадь шкуры неубитого слона. Известно, что его шкура имеет форму прямоугольника $A \times B$ метров.

Формат входных данных

В единственной строке ввода записаны через пробел числа A и B.
$1 \leqslant A \times B \leqslant A + B$, $1 \leqslant A, B \leqslant 10^{100\,000}$.

Формат выходных данных

В единственной строке выведите площадь шкуры неубитого слона.

Примеры

stdin	stdout
1 1	1

Задача F. Повторите, я вас не слышу!

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

На лекции школьники группы D0 сидят за стульями, расставленными в n рядов по n штук. Ряды и места в рядах задаются числами от 1 до n. Каждое место характеризуется слышимостью — количеством слов, которые школьник сумел услышать за время лекции. Слышимости всех мест различны.

За обедом преподаватели решили выяснить, есть ли школьник, которому слышно хуже, чем всем его соседям. Проблема в том, что школьники уже разбежались, и поймать их непросто. Преподаватели умеют за одну минуту находить школьника и узнавать у него, сколько слов он услышал.

У школьника не более восьми соседей: они сидят слева, справа, спереди, сзади и по четырём диагоналям диагоналям от него.

Формат входных данных

В первой строке будет введено целое число n — количество стульев в рядах. $3 \leqslant n \leqslant 500$.

Далее ваша программа может запрашивать значение слышимости на конкретном стуле. Запрос представляет собой пару чисел i, j. В ответ на запрос на стандартный ввод ваша программа получит целое число — значение слышимости на стуле в ряду i с номером j.

Разрешается сделать не более $10 \times n$ запросов. После последнего запроса следует вывести пару чисел $0, 0$, а в следующей строке — ответ. Ответ также представляет собой пару чисел: номер ряда и номер стула в ряду, такого, что все его соседи имеют большее значение слышимости.

После каждой строки, выведенной вашей программой, вызывайте функцию сброса буфера вывода:

- Pascal: `flush(output)`
- C: `fflush(stdout)`
- C++: `cout.flush()`
- Java: метод `flush()` вашего `PrintWriter` или аналогичного объекта
- Python: добавьте `flush=true` в параметры `print`

Формат выходных данных

Примеры

stdin	stdout
3 1 2 3 4 5	1 1 1 2 1 3 2 1 2 2 0 0 1 1

Задача G. Старая фотография

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Старая фотография, которую фотограф-любитель Юрий нашёл у себя на дискете, представляет собой черный прямоугольник с несколькими белыми пикселями, причем белые пиксели образуют прямоугольный многоугольник, стороны которого параллельны осям координат. Прямоугольный многоугольник - это многоугольник без самопересечений и самокасаний, у которого все углы прямые.

«Это подозрительно похоже на фотографию белого прямоугольника», — подумал Юрий.

А какое минимальное отличие у этой фотографии и у самой похожей на нее фотографии вида: белый прямоугольник со сторонами параллельными сторонам фотографии на черном фоне? Под отличием двух фотографий понимается количество пикселей, отличающихся при наложении.

Формат входных данных

В первой строке содержатся одно число — количество тестов t. Каждый тест состоит из строки, в которой содержится число n ($4 \leqslant n \leqslant 800$) — количество вершин белого многоугольника, и n строк, в которых содержатся по два числа, — координаты вершин белого многоугольника. Сумма n по всем тестам не превосходит 800.

Все координаты не превышают 1 000 000 по абсолютной величине.

Формат выходных данных

Для каждого теста выведите в отдельной строке минимальное отличие данной фотографии от фотографии вида белый прямоугольник на черном фоне.

Примеры

stdin	stdout
2 10 3 1 4 1 4 3 3 3 3 4 1 4 1 3 2 3 2 2 3 2 4 0 0 1 0 1 1 0 1	3 0

Задача H. Кольцевая–Шмольцевая

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

2345-й год. Кольцевая линия Судиславского метрополитена имеет форму окружности. Шмольцевая линия — форму квадрата. Линии нигде не пересекаются, поэтому необходимо найти местоположение для строительства вестибюля станции «Кольцевая–Шмольцевая» так, чтобы расстояние от него до обеих линий было одинаковым.

Расстоянием от точки p до фигуры A назвыается минимальное из расстояний от p до всех точек q, принадлежащих A.

Известно, что Шмольцевая линия не может располагаться внутри Кольцевой.

Формат входных данных

В первой строке входного файла заданы целые числа $r > 0$, x_1, y_1, x_2, y_2 — радиус Кольцевой линии и координаты противоположных углов Шмольцевой линии.

Все числа не превышают 10^4 по абсолютной величине.

Формат выходных данных

Выведите координаты любой точки, где можно построить вестибюль «Кольцевой–Шмольцевой».

Расстояния от выведенной Вами точки до линий метрополитена должны совпадать (относительная или абсолютная погрешность не превосходит 10^{-6}).

Примеры

stdin	stdout
1 2 3 6 2	0.0 3.0

Задача I. Всё, что тебя касается

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

На плоскости расположена окружность, но мы не скажем где.
На плоскости расположена точка, но мы не скажем где.
Вам предлагается построить из исходной точки касательную к окружности.

Формат входных данных

Это интерактивная задача.
Ниже представлен протокол общения решения и проверяющей системы.
Во всех запросах идентификатор объекта — это строка из заглавных букв латинского алфавита. Длина идентификатора в точности равна четырём.
В первой строке ваша программа получает идентификатор окружности в формате «`CIRCLE: ID`».
Во второй строке ваша программа получает идентификатор точки, касательную из которой нужно построить: «`POINT: ID`».
Далее ваша программа может выполнять следующие запросы:

- `TOUCH ID` — возвращает случайную точку объекта ID (окружности или прямой), которой ещё не было.
- `LINE IDA IDB` — возвращает имя прямой, проходящей через точки IDA и IDB
- `INTERSECT IDA IDB` — пересекает объекты IDA и IDB и возвращает точки пересечения (или прямую, если пересекаются две совпадающие).

В ответ на каждый из этих запросов ваша программа получает на стандартный ввод строку, содержащую от нуля до двух идентификаторов, завершённую переводом строки, в формате `{ID1, ID2, ..., IDN}`

- `TANGENT ID` — сообщить, что указанная прямая — искомая касательная. Запрос используется один раз непосредственно перед прекращением выполнения программы.

Ваша программа может сделать не более тридцати запросов.
После каждой строки, выведенной вашей программой, вызывайте функцию сброса буфера вывода:

- Pascal: `flush(output)`
- C: `fflush(stdout)`
- C++: `cout.flush()`
- Java: метод `flush()` вашего `PrintWriter` или аналогичного объекта
- Python: добавьте `flush=true` в параметры `print`

Формат выходных данных

Примеры

stdin	stdout
CIRCLE: WWWW POINT: PPPP {AAAA} {BBBB} {LLLL} {AAAA, QQQQ} {LLLL}	TOUCH WWWW TOUCH WWWW LINE AAAA PPPP INTERSECT LLLL WWWW LINE PPPP QQQQ TANGENT LLLL

Замечание

Вам представлен пример общения решения с проверяющей системой. Обратите внимание, что в результате представленного взаимодействия будет получен неверный ответ, так как прямая `LLLL` пересекает окружность `WWWW` в двух точках: `AAAA` и `QQQQ`

Задача J. Борода админа

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Известный спелеолог, автор многотомного фолианта по спелеологии, Дональд К. исследовал пещеру «Борода админа». Пещера называется так, потому что похожа на бороду админа следующим свойством: от любого грота до любого другого существует ровно один простой путь по лазам. (С этого момента будем называть гроты — вершинами, лазы — ребрами, а пещеру — деревом, причем подвешенным за первую вершину. Надеемся, Дональд К. сможет разобраться в нашей терминологии.)

Дональд К. использовал следующую процедуру обхода пещеры, начиная с грота, в котором расположен вход, то есть с первой вершины:

- построить предположение о том, сколько вершин в поддереве этой вершины, записать в блокнот,
- для каждой из ещё не посещенных вершин, соседних с данной,
- пойти в нее, применить данную процедуру, вернуться,
- убедиться, что предположение было верным: Дональд К. не ошибается никогда.

Вы нашли блокнот с записями Дональда К. Восстановите план пещеры «Борода админа».

Формат входных данных

В первой строке содержится количество прогнозов Дональда К. $N \leqslant 100\,000$. Во второй строке содержатся прогнозы Дональда К. — натуральные числа, не превосходящие $100\,000$.

Формат выходных данных

Выведите дерево, в котором описанная процедура дает такие результаты. А именно, необходимо вывести $N - 1$ число: для каждой вершины v, кроме стартовой, необходимо вывести номер той вершины, из которой Дональд К. попал в вершину v впервые.

Если таких деревьев несколько, выведите любое.

Гарантируется, что хотя бы одно такое дерево существует.

Примеры

stdin	stdout
5 5 1 3 1 1	1 1 3 3