Задача A. Машинка

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вася играет в следующую компьютерную игру. На бесконечной плоскости в начале координат стоит машинка. За один ход машинка может повернуть налево или направо на 90 градусов, а может не менять направления движения; после этого она перемещается на расстояние 1 в получившемся направлении. Изначально машинка повёрнута в направлении точки $(0,\ 1)$. На ход тратится одна секунда.

Ходы Васи записываются по порядку. Каждому повороту направо соответствует буква `A`, каждому перемещению вперёд на расстояние 1 — буква `B`, а каждому повороту налево — буква `C`.

Цель игры — попасть в заданную точку $(x,\ y)$; при этом в зависимости от получившейся записи пути игроку начисляется некоторое количество очков. Уровень сложности, выбранный Васей, таков, что машинка должна приехать в пункт назначения за минимально возможное время.

Проехав очередным зигзагом, Вася получил сообщение-подсказку: если проехать по лексикографически k-му из кратчайших по времени путей, то откроется секретный уровень! Вася знает, что такое лексикографический порядок: строка $\mathcal{S} = s_1 s_2 \dots s_m$ лексикографически меньше строки $\mathcal{T} = t_1 t_2 \dots t_n$, если либо $m < n$ и $s_i = t_i$ для всех i от 1 до m, либо для какого-то $l \leqslant \min(m,\ n)$ первые $(l-1)$ символов строк $\mathcal{S}$ и $\mathcal{T}$ совпадают, а $s_l < t_l$. Так что дело за малым — всего лишь выписать все возможные строки, соответствующие кратчайшим путям, в этом порядке, от меньших к большим, найти k-ю из них и повторить путь, описываемый этой строкой...

К сожалению, строк может быть очень много, и Вася будет слишком долго выписывать и сортировать их вручную. Помогите ему найти лексикографически k-ю строку изо всех строк, соответствующих кратчайшим по времени путям!

Формат входного файла

В первой строке входного файла записаны целые числа x и y через пробел — координаты пункта назначения $(0 \leqslant x,\ y \leqslant 100)$. Во второй строке записано натуральное число k — номер пути, по которому надо проехать, чтобы попасть на секретный уровень. Известно, что пункт назначения не совпадает с началом координат, а лексикографически k-й путь существует.

Формат выходного файла

Выведите в выходной файл одну строку — запись лексикографически k-го пути из начала координат в точку $(x,\ y)$. Строка должна содержать лишь символы `A`, `B` и `C`.

Примеры

standard input	standard output
2 1 1	ABBCB
2 1 2	ABCBAB
2 1 3	BABB

Пояснение к примеру

Существует всего три пути из $(0,\ 0)$ в $(2,\ 1)$ за минимальное время — три секунды. Они записываются как `ABBCB` (путь $(0,\ 0)$ – $(1,\ 0)$ – $(2,\ 0)$ – $(2,\ 1)$), `ABCBAB` (путь $(0,\ 0)$ – $(1,\ 0)$ – $(1,\ 1)$ – $(2,\ 1)$) и `BABB` (путь $(0,\ 0)$ – $(0,\ 1)$ – $(1,\ 1)$ – $(2,\ 1)$). В лексикографическом порядке `ABBCB` < `ABCBAB` < `BABB`.

Задача B. Почти полный граф

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Граф называется *почти полным*, если для любых трёх различных вершин в нём есть хотя бы два ребра между какими-то из этих вершин. Требуется проверить, является ли данный граф почти полным. В графе нет петель и кратных рёбер.

Поскольку граф может быть очень большим, он задан не матрицей смежности и даже не списком рёбер, а следующим кодом. Пусть $a_{i,j}$ равно единице, если между вершинами i и j есть ребро, и нулю в противном случае. Выпишем подряд числа $a_{1,2}$, $a_{1,3}$, ..., $a_{1,n}$, $a_{2,3}$, $a_{2,4}$, ..., $a_{2,n}$, $a_{3,4}$, ..., $a_{n-1,n}$ (всего у нас получится $\frac{n(n-1)}{2}$ чисел). Теперь разобьём эту последовательность на группы одинаковых чисел, идущих подряд. Закодируем её так: сначала запишем s — первое число (0 или 1), затем — размер первой группы t_1 (символы в начале последовательности, совпадающие с s), затем — размер второй группы t_2 (группа одинаковых символов после первых t_1, не совпадающих с s), затем — размер третьей группы t_3 (группа одинаковых символов после первых t_1+t_2, совпадающих с s), и так далее до t_l — размера последней группы одинаковых символов.

Формат входного файла

В первой строке входного файла заданы два целых числа n и l через пробел — количество вершин графа и количество групп в последовательности, задающей граф ($2 \leqslant n \leqslant 50\,000$, $1 \leqslant l \leqslant 50\,000$). Во второй строке записаны через пробел целые числа $s\ t_1\ t_2\ \ldots t_l$; s равно 0 или 1, все t_i положительны и сумма $t_1 + t_2 + \ldots + t_l$ равна $\frac{n(n-1)}{2}$.

Формат выходного файла

Выведите в первую строку выходного файла слово '`YES`', если данный граф является почти полным, и '`NO`' в противном случае. Кроме того, если ответ — '`NO`', то во второй строке выведите три числа — номера трёх вершин графа, между которыми нет хотя бы двух рёбер.

Пример

standard input	standard output
6 4 0 1 4 5 5	NO 2 3 4

Пояснение к примеру

Матрица смежности данного графа имеет следующий вид:

$$\begin{array}{cccccc} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{array}$$

Очевидно, что этот граф не является почти полным: например, между вершинами 2, 3 и 4 (считая с единицы) нет ни одного ребра, а требуется, чтобы их было не менее двух.

Задача C. Сравнение строк

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Циклическое расширение $\mathcal{S}^*$ строки $\mathcal{S}$ — это строка $\mathcal{S}$, приписанная сама к себе бесконечное количество раз; к примеру, циклическим расширением строки ‘bab’ является строка ‘babbabbab...’.

Даны длины двух строк $\mathcal{S}$ и $\mathcal{T}$ — числа m и n. Рассмотрим циклические расширения $\mathcal{S}^*$ и $\mathcal{T}^*$ этих строк. Как проверить, равны ли они?

Наивный алгоритм будет проверять строки на равенство, просто сравнивая s_1 с t_1, s_2 с t_2 и так далее. В итоге алгоритм либо найдёт пару несовпадающих символов, либо, если строки равны, будет проводить сравнения бесконечно долго.

Однако понятно, что, если строки не равны, то последняя позиция p, на которой мы можем встретить различие ($s_p \neq t_p$), конечна. Мы хотим узнать, чему равно p, чтобы улучшить алгоритм сравнения так: новый алгоритм будет сравнивать символ s_1 с t_1, s_2 с t_2 и так далее, пока либо не найдёт несовпадение $s_q \neq t_q$ на позиции $q \leqslant p$, либо, сравнив первые p пар и обнаружив соответствие символов в каждой паре, не докажет тем самым равенство строк $\mathcal{S}^*$ и $\mathcal{T}^*$.

Для данных длин строк m и n приведите пример строк $\mathcal{S}$ и $\mathcal{T}$ соответствующей длины, циклические расширения которых различны, но первое несовпадение встречается как можно позже.

Формат входного файла

В первой строке входного файла заданы два числа через пробел — это числа m и n ($1 \leqslant m, n \leqslant 100$).

Формат выходного файла

Выведите в первую строку входного файла строку $\mathcal{S}$, а во вторую — строку $\mathcal{T}$. Строки могут содержать только строчные буквы латинского алфавита (‘a’–‘z’).

Система оценки

Ответ на каждый тест оценивается следующим образом. Если приведённые строки некорректны или их циклические расширения совпадают, ответ оценивается в 0 баллов. Если же ответ на тест корректен, то место первого несовпадения q сравнивается с максимально возможным значением q для данных m и n — величиной p, и участник получает $B \cdot 2^{q-p}$ баллов, где B — максимальное количество баллов за этот тест. Баллы за каждый тест суммируются, и итоговая сумма округляется к ближайшему целому числу.

Жюри гарантирует, что существует решение, находящее ответ с максимально возможным значением q на любом корректном тесте и укладывающееся в ограничения по времени и памяти.

Пример

standard input	standard output
2 4	aa aaab

Пояснение к примеру

Для $m = 2$ и $n = 4$ значение p равно четырём, и оно достигается на строках ‘aa’ и ‘aaab’, циклические расширения которых — строки ‘aaaa...’ и ‘aaab...’ — различаются в четвёртом символе. Если ответ участника на этот тест — строки ‘aa’ и ‘aabc’, то первое несовпадение получается на третьей позиции, и такой ответ даст лишь половину баллов за тест. Если же ответ участника — строки ‘yz’ и ‘yzyz’, то циклические расширения равны, и количество баллов равно нулю.

Задача D. Гирлянда

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Новогодняя гирлянда, установленная за окном Васиного дома, представляет из себя n лампочек, расположенных на одной прямой и пронумерованных подряд числами от 0 до $(n-1)$. Вася, глядя в окно, заинтересовался порядком включения лампочек. После внимательных наблюдений он установил следующее.

Изначально все лампочки выключены. Далее каждую секунду включается одна лампочка. В первую секунду включается лампочка с номером a, во вторую — с номером $a^2 \mod n$, в третью — с номером $a^3 \mod n$, ..., в k-ю — лампочка с номером $a^k \mod n$.

Вася выяснил значения чисел a и n, и теперь его заинтересовал следующий вопрос. Рассмотрим лампочку, включённую в r-ю секунду (она имеет номер $a^r \mod n$). Сколько лампочек слева от неё, то есть с меньшими номерами, уже включены? Вася считает, что именно свойства этих чисел для разных значений r и обеспечивают гирлянде особую красоту. Однако сам он будет слишком долго вручную их вычислять. Помогите Васе — в ответ на каждый заданный им вопрос о числе r выведите количество лампочек, зажжённых до r-й зажжённой лампочки и находящихся слева от неё.

Формат входного файла

В первой строке входного файла записаны целые числа a, n и q через пробел ($1 \leqslant a < n \leqslant 1\,000\,000$, $1 \leqslant q \leqslant 10\,000$). Следующие q строк содержат по одному целому числу r_i каждая ($1 \leqslant r_i \leqslant 100\,000$) и описывают Васины вопросы. Вася не знает, что будет, когда по этому алгоритму надо будет зажечь уже зажжённую лампочку, поэтому его вопросы таковы, что для любого i за первые r_i секунд никакая лампочка не должна будет загореться дважды.

Формат выходного файла

Выведите в выходной файл q строк, по одному числу в каждой строке. В i-й строке выведите количество лампочек, зажжённых раньше r_i-й зажжённой лампочки и имеющих меньшие номера.

Примеры

standard input	standard output
2 3 2 1 2	0 0
5 16 4 1 3 4 2	0 2 0 1
2 8 2 2 2	1 1

Пояснение к примеру

В первом примере порядок зажигания лампочек — 2, 1.

Во втором примере — 5, 9, 13, 1.

В третьем примере — 2, 4, 0.

Задача E. Подобные прямоугольники

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Даны n точек на плоскости; абсциссы всех точек попарно различны, так же как и все ординаты. Для каждой пары различных точек рассмотрим прямоугольник со сторонами, параллельными осям координат, в котором они являются противоположными углами. Существует ли такой набор этих прямоугольников, содержащий не менее трети всех получившихся прямоугольников, что все прямоугольники в этом наборе подобны?

Формат входного файла

В первой строке входного файла содержится целое число n — количество точек ($2 \leqslant n \leqslant 1000$). В следующих n строках записано по два целых числа x_i y_i в каждой через пробел — координаты точек. Все x_i различны, все y_i различны, и координаты не превосходят 10 000 по абсолютной величине.

Формат выходного файла

Если искомый набор существует, выберите любой прямоугольник из любого такого набора и в первой строке выведите k — количество прямоугольников, подобных ему, включая его самого, а во второй строке выведите через пробел номера точек, являющихся вершинами этого прямоугольника, в любом порядке. Если же это неверно, выведите число -1 в первой строке выходного файла. Точки нумеруются с единицы в том порядке, в котором они даны во входном файле. Помните, что при положительном ответе k должно быть не меньше, чем $\frac{1}{3} \times \frac{n(n-1)}{2}$.

Примеры

standard input	standard output
5 1 1 2 2 3 3 4 5 5 6	4 4 5
4 0 0 1 4 9 16 25 36	-1

Задача F. Замощение

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вася заигрался в игру, описанную в предыдущей задаче, и совсем забыл о том, что набирал воду в ванну. Конечно же, вода вылилась наружу! В результате в ванной комнате пострадал прямоугольный участок пола размера $m \times n$ квадратных дециметров. Вася стал искать, чем бы его заделать, и нашёл в чулане плитки двух типов — квадратные размера 4×4 квадратных дециметра и прямоугольные размера 3×6 квадратных дециметров. Вася решил замостить повреждённый участок пола этими плитками. Оказалось, что прямоугольник $m \times n$ можно замостить такими плитками, и плиток обоих типов достаточно для любого возможного замощения. Однако у Васи мало замазки, чтобы заделывать щели между плитками, да и выглядят щели некрасиво. Поэтому он решил выбрать такое замощение, чтобы минимизировать суммарную длину всех получившихся щелей. Теперь Вася хочет узнать эту минимальную суммарную длину, чтобы понимать, как расходовать замазку. Какова эта длина?

Формат входного файла

В первой строке входного файла записаны целые числа m и n через пробел — размеры повреждённого участка пола ($1 \leqslant m, n \leqslant 20$). Данные m и n таковы, что замощение возможно.

Формат выходного файла

Выведите в выходной файл одно число — минимальную суммарную длину всех щелей, получающихся при замощении, в дециметрах.

Примеры

standard input	standard output
4 4	0
3 12	3
9 6	12