Задача A. Алфавит

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

Воспитательница Галя работает в детском саду. Кроме детских игр, в этом детском саду проходят занятия. Вот уже неделю ребята изучают буквы латинского алфавита.

Каждое утро воспитательница выстраивает всех своих подопечных в ряд, и они играют в игру. Первый ребёнок в ряду громко называет первую букву алфавита — A. Второй должен назвать B, третий — C, и так далее. По счастливому стечению обстоятельств, всего в группе 26 детей — столько же, сколько и букв в латинском алфавите.

Если каждый ребёнок без ошибки назовёт свою букву, то группа отпразднует знание латинского алфавита, и ребята по этому случаю съедят большой торт. Однако пока что группе не удаётся правильно назвать все 26 букв — каждое утро то один, то другой называет свою букву неправильно, и игра на этом заканчивается.

Ребята учат буквы подряд, и каждый из них знает первые несколько букв алфавита и не может назвать остальные. Поэтому возможность выиграть напрямую зависит от их расстановки. К примеру, если последним в ряду окажется ребёнок, знающий латинский алфавит только до буквы M, то букву Z он назвать не сможет, и группа не выиграет независимо от того, насколько хорошо выучили алфавит остальные ребята.

Галя считает, что группа в целом уже достаточно хорошо знает алфавит, и хочет помочь своим ребятам выиграть. Для этого ей нужно всего лишь расставить их так, чтобы первый ребёнок в ряду знал алфавит хотя бы до буквы A, второй — хотя бы до буквы B, и так далее; последний в ряду должен знать все буквы.

Помогите Гале решить, в каком порядке расставить ребят, чтобы они смогли выиграть, или выясните, что это пока невозможно.

Формат входного файла

В первой строке входного файла записана строка из 26 заглавных букв латинского алфавита; i-ая из этих букв говорит о том, до какой буквы включительно выучил алфавит i-ый ребёнок.

Формат выходного файла

В первой строке выходного файла выведите "`YES`", если воспитательнице удастся выстроить своих детей в ряд так, чтобы они выиграли, и "`NO`", если им для этого ещё надо поучиться. Если расстановка возможна, во второй строке выведите перестановку из 26 чисел от 1 до 26 через пробел — порядок детей в ряду. Если решений несколько, можно вывести любое.

Примеры

abc.in
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abc.out
YES 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

abc.in
BCARTYXYZZYZZYXYXYZZYZZYXV
abc.out
YES 3 1 2 4 5 26 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 25 24 23

abc.in
AAZZZZZZZZZZZZZZZZZZZZZZZZ
abc.out
NO

Задача B. Графы

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

Однажды в древнее государство Оссия приехал учёный японец Хисикоши. Он обнаружил, что в этом государстве есть одна большая проблема — дороги, а точнее, их отсутствие. Ещё он обнаружил другую проблему — графов (а было их N), которые, мягко говоря, не отличались особенным уровнем интеллекта. Хисикоши решил помочь построить дороги. С рабочей силой проблем не было, поэтому он решил подойти к этому мероприятию с выдумкой. Он узнал имена всех графов, и записал их по-японски (а в Японии, как известно, пользуются иероглифами). Он решил, что надо провести дорогу между замками двух графов (и ввести на ней одностороннее движение от первого ко второму — чтобы интереснее было), если последняя буква имени первого графа совпадает с первой буквой имени второго. Однако оказалось, что не от каждого графа можно доехать до всех других. Тогда он решил поселить еще несколько графов и дать им имена так, чтобы теперь можно было бы доехать от любого графа до любого другого. Какое наименьшее количество графов ему надо добавить?

Будем считать, что японский иероглиф однозначно кодируется тремя латинскими буквами. И каждой трёхбуквенной комбинации соответствует свой иероглиф.

Формат входного файла

В первой строке находится целое число N — количество графов ($1 \leqslant N \leqslant 100\,000$). В следующих N строках написаны их имена. При этом гарантируется, что длина каждого имени (в латинских символах) делится на 3. Имена написаны большими латинскими буквами. Каждое имя содержит от одного до десяти иероглифов.

Формат выходного файла

Выведите единственное число — минимальное количество новых графов, которое должен поселить Хисикоши.

Примеры

counts.in	counts.out
2 AAABBB BBBCCC	1
2 AAABBB CCCDDD	2
3 AAABAA ABAAAB AABBAA	2

Задача C. Покрытие

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

Представим себе комнату размера $m \times n$, пол которой расчерчен линиями сетки на квадраты 1×1. Предлагается покрыть пол этой комнаты дощечками весьма необычного вида.

Одна дощечка представляет собой фигуру, состоящую ровно из 6 клеток квадрата 3×3 и являющуюся связной по стороне (иными словами, из любой клетки можно попасть в любую другую, перемещаясь между центрами соседних клеток только по вертикали и горизонтали и не покидая пределов фигуры). Все дощечки одинаковы. Дощечки выкладываются на пол так, чтобы стороны их клеток совмещались с линиями сетки на полу; их нельзя ни поворачивать, ни перевернуть и положить обратной стороной вверх.

Покрытием назовём такое положение дощечек, что каждая клетка пола покрыта хотя бы одной дощечкой из этого набора. Покрытие клетки пола более чем одной дощечкой, равно как и покрытие клеток вне пределов комнаты, допускается.

Требуется узнать, какое минимальное количество дощечек данного вида потребуется, чтобы построить покрытие пола нашей комнаты.

Формат входного файла

В первой строке входного файла заданы через пробел два целых числа m и n ($1 \leqslant m, n \leqslant 9$). В следующих трёх строках описан квадрат 3×3, содержащий дощечку. Каждая из этих строк содержит ровно три символа. Символ '`X`' (икс большое) соответствует клетке дощечки, а символ '`.`' (точка) — пустой клетке. Гарантируется, что эти строки не содержат других символов, суммарное количество символов '`X`' на них равно шести, и фигура из букв '`X`' связна по стороне.

Формат выходного файла

Выведите в выходной файл одно число — минимальное количество дощечек данного вида, которые нам потребуются, чтобы построить покрытие пола.

Примеры

covering.in	covering.out
3 3 .X. XXX XX.	3
3 2 XXX XXX ...	1
2 3 XXX XXX ...	2
2 2 XXX X.X ..X	2

Задача D. Фатализм

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

Роботу Прайму, Конструктору Роботов, надоели его творения — они получаются слишком тупыми и бездумными, и среди них не найти собеседника и даже просто мыслителя, достойного общаться с Ним, Праймом, Великим И Неповторимым. Прайм испробовал уже все известные ему методы, но создать себе подобного до сих пор не получилось. В отчаянии он придумал себе игру, чтобы как-то отвлечься от своей неразрешимой проблемы, преодолеть творческий кризис и заодно пустить побольше своих творений по правильному пути. Игра называется «Фатализм» и заключается в следующем.

Прайм создаёт лабиринт размера $m \times n$ клеток, огороженный со всех сторон стенами. Каждая клетка лабиринта — либо свободное пространство, либо стена. В начальный момент времени в некоторых клетках, где нет стены, стоят роботы и смотрят в одном из четырёх направлений. Никакие два робота не стоят в одной клетке.

Каждый робот движется с постоянной скоростью — одна клетка в секунду — в направлении, в котором он смотрит, и светит лазером в этом же направлении до ближайшей стены. В конце каждой секунды некоторые роботы взрываются. Это происходит в трёх случаях.

- Если робот оказался за пределами лабиринта или в клетке, которая заполнена стеной, то он взрывается.
- Если в какой-то клетке оказалось два или более робота, то все они взрываются.
- Для оставшихся роботов, если какой-то из них оказался между другим роботом и стеной, до которой светит луч его лазера, то этот робот взрывается. Взрывы от лучей лазера происходят одновременно; тем не менее, даже если какой-то робот взорвался, луч его лазера успевает взорвать всех роботов, которые находились между ним и стеной, до которой светит его лазер.

Известно, что в начальный момент времени никакой робот не светит на другого робота лазером. Взорвавшиеся роботы не оказывают влияния на ситуацию в лабиринте в последующие секунды.

Понятно, что жизнь любого робота будет в таких условиях очень недолгой. Прайм хочет выяснить, сколько времени просуществует самый долгоживущий из них.

Выясните, сколько времени пройдёт до того момента, как все роботы взорвутся.

Формат входного файла

В первой строке входного файла заданы через пробел три числа m, n и k ($1 \leqslant m, n, k \leqslant 100$). В следующих n строках содержится ровно по m символов в каждой; i-ый символ в j-ой из этих строк равен '`X`' (икс большое), если соответствующая клетка (i, j) занята стеной, и '`.`' (точка), если эта клетка пуста. Далее идут k строк, описывающие роботов. Каждая из них имеет вид x_i y_i z_i, где x_i и y_i — координаты робота ($1 \leqslant x_i \leqslant m$, $1 \leqslant y_i \leqslant n$), а z_i — один из четырёх символов направления: символ '`U`' (up) соответствует уменьшению координаты y, символ '`L`' (left) — уменьшению координаты x, символ '`D`' (down) — увеличению y, а символ '`R`' (right) — увеличению x. Все числа во входном файле целые.

Формат выходного файла

Выведите в выходной файл одно число — сколько секунд пройдёт до того момента, как все роботы взорвутся.

Примеры

fatalism.in	fatalism.out
5 1 2 X...X 3 1 L 4 1 R	2
4 4 4 X..X .... X..X 1 3 R 3 4 U 4 2 L 2 1 D	1
4 5 3 .... 1 4 R 3 5 U 4 5 U	4
3 6 5 .X. .X. .X. ... ... .X. 1 4 R 2 5 U 3 5 U 1 6 R 3 6 L	5

В первом примере роботы изначально смотрят в разные стороны и взрываются, врезавшись в противоположные стены.

Во втором примере в конце первой секунды роботы сжигают друг друга по кругу своими лазерами.

В третьем примере в конце первой секудны первый робот сжигает лазером двух других, а оставшись один, в конце четвёртой секунды выходит за пределы лабиринта и взрывается сам.

В четвёртом примере первые два робота сталкиваются и взрываются в конце первой секунды, и лазер первого робота не успевает никого сжечь; четвёртый и пятый роботы просто врезаются в стену между ними, а третий движется вверх, пока не взорвётся, покинув пределы лабиринта.

Задача E. Губернатор

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

Вам, как губернатору города, необходимо организовать в своём городе постройку нескольких зданий, чтобы привести его в соответствие мировым стандартам. Губернатор вы не простой, а с высшим экономическим образованием, поэтому вас в первую очередь заботит денежная сторона вопроса. Создав комиссию по этому поводу и проведя первое заседание, вы выяснили следующее.

В настоящее время, то есть до постройки требуемых зданий, город приносит стабильный доход — K золотых монет в месяц. Однако постройка любого здания может существенно изменить сложившееся положение.

Пронумеруем здания, требующие постройки, числами от 1 до N, где N — их количество. Каждое здание i характеризуется двумя числами. Во-первых, это число a_i — его эффективность. Она является вещественным числом и обозначает то, во сколько раз возрастёт текущая прибыль при постройке этого здания. Второй параметр b_i — это количество золотых монет, которые каждый месяц уходят на содержание этого здания. В итоге, если к моменту постройки i-ого здания месячный доход составлял X монет, то после постройки этого здания его величина станет равна $a_i \cdot X - b_i$. Отметим, что итоговый доход не обязательно будет целым числом.

Следует также учесть, что комитет по городскому строительству имеет сравнительно небольшой и малообученный штат. Нехватка квалифицированных административных кадров проявляется прежде всего в том, что строители не могут работать над двумя проектами одновременно, а начав работу над одним зданием, не могут перейти к другому, не закончив первое.

Перед комиссией теперь стоит нелёгкая задача: решить, в каком порядке строить здания — а строить их надо все, даже те, которые невыгодны — чтобы получать максимальный доход после постройки всех зданий. Как и следовало ожидать, для ваших коллег это оказалось слишком сложной задачей. Помогите им!

Формат входного файла

В первой строке входного файла записаны через пробел два целых числа N и K ($1 \leqslant N \leqslant 10\,000$, $1 \leqslant K \leqslant 10^6$). Далее в каждой из N последующих строк описано по зданию. Каждое описание имеет вид a_i b_i, где число a_i — вещественное ($0 \leqslant a_i \leqslant 10$), а b_i — целое ($1 \leqslant b_i \leqslant 100$). i-ая из строк файла соответствует $(i-1)$-ому зданию в нумерации, используемой комиссией.

Формат выходного файла

Выведите в выходной файл перестановку из N чисел от 1 до N, по одному числу на строку — номера зданий в порядке их постройки. Если существует несколько перестановок, максимизирующих прибыль, разрешается вывести любую.

Примеры

governor.in	governor.out
2 10 1 5 2 3	2 1
4 6 1.2 3 1.5 2 2.0 4 0.5 1	2 3 1 4

Задача F. Самое экстравагантное дупло

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

> Не долбить дятел не может. Потому что клюв всегда перевешивает.
>
> Е. В. Шестаков, «Крылья Родины»

В Звенящем Лесу проводится конкурс на самое экстравагантное дупло. Правда, в целях экономии нервов зрителей, конкурс проводится среди макетов, а не самих дупел.

В качестве исходного материала каждому из участников предоставляется доска размером $20\,000 \times 20\,000$ метров. Макет элементарного дупла представляет из себя окружность, аккуратно выдолбленную на плоскости. Макет же экстравагантного дупла представляет из себя несколько макетов элементарных дупел, выдолбленных на одной доске.

В целях неуменьшения количества справедливости в мире, каждому участнику уже заданы радиусы всех элементарных дупел, которые он должен выдолбить. Более того, даже последовательность их выдалбливания строго зафиксирована. Таким образом, единственное, что каждый из участников может варьировать — это координаты центра каждого элементарного дупла.

Казалось бы, все возможности для свободного творчества уже перекрыты, но это не так. Во-первых, каждому элементарному дуплу соответствует некоторое изначальное количество баллов. Во-вторых, сразу после выдалбливания очередного элементарного дупла за него начисляется итоговый балл, равный сумме изначального балла за это дупло и всех итоговых баллов за уже выдолбленные элементарные дупла, с которыми данное дупло пересекается. Итоговый балл за экстравагантное дупло начисляется как сумма итоговых баллов за все элементарные дупла, из которых оно состоит. Два дупла считаются пересекающимися, если соотвествующие им на макете окружности пересекаются, либо одна лежит внутри другой. Касания окружностей на макете недостаточно, чтобы считать дупла пересекающимися.

Вам предлагается примерить на себя шкуру дятла и написать программу, которая по заданным радиусам элементарных дупел, баллов за них, а также последовательности их выдалбливания определила бы координаты центров элементарных дупел, да так, чтобы получившееся экстравагантное дупло имело бы максимальный возможный итоговый балл.

Формат входного файла

В первой строке входного файла задано число n — количество элементарных дупел, которые надо задействовать ($1 \leqslant n \leqslant 1\,000$). Далее следуют n строк, каждая из которых описывает соответствующее дупло и содержит два целых числа: r_i и c_i ($0 < r_i$, $c_i \leqslant 1000$), где r_i — радиус соответствующего дупла в метрах, а c_i — изначальное количество баллов, соответствующее данному элементарному дуплу. Дупла даны в порядке выдалбливания.

Формат выходного файла

Выходной файл должен содержать ровно n строчек, в каждой по два числа — координаты центра соответствующего элементарного дупла в метрах с точностью не менее 10^{-6} метра. Кроме того, макеты дупел не должны выходить за пределы исходной доски, а расстояние между любыми двумя центрами должно быть не менее сантиметра. Точка $(0, 0)$ соответствует центру исходной доски. Если возможны несколько вариантов расстановки дупел, дающих максимальный балл, выведите любой.

Пример

hollow.in	hollow.out
3 10 10 20 20 30 30	0 1 0 2 0 3

Задача G. Дорога

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

В Древнем государстве Оссия было два города, между которыми была проложена дорога длиной S метров. Через каждый метр стояли столбики, на каждом из которых по некоторому принципу (этот секретный принцип был известен только древним монахам Шамбалы) было написано по букве (а алфавит там у них был латинский). Однажды князь-король Василий I решил, что человек, когда он едет по этой дороге, слишком редко вспоминает о нём. Он решил это исправить. Для этого он повелел на некоторых столбиках вместо буквы написать «Здесь был Вася». По его представлению, человек, проехав любой участок дороги длиной K метров, должен обязательно хоть раз увидеть такую надпись. Иными словами, среди каждых K идущих подряд столбиков должен оказаться хоть один, на котором буква заменена на надпись. При этом, чтобы не слишком раздражать монахов (а они люди обидчивые), Василий I приказал выбрать для надписи такие столбики, чтобы среди стёртых букв оказалось как можно меньше различных букв латинского алфавита.

Помогите боярам выполнить приказ своего повелителя.

Формат входного файла

В первой строке написано одно целое число K ($1 \leqslant K \leqslant 100\,000$). Во второй строке — без пробелов написано S заглавных латинских букв в той последовательности, в которой ими помечены столбики вдоль дороги. Гарантируется, что $K \leqslant S \leqslant 100\,000$.

Формат выходного файла

В первой строке выведите N — минимальное количество различных букв латинского алфавита, которые хотя бы на одном столбике придётся стереть, чтобы написать «Здесь был Вася». В следующих N строках выведите те заглавные буквы латинского алфавита, которые потребуется хоть раз стереть. Буквы можно выводить в любом порядке. Если ответов с минимальным N несколько, можно вывести любой из них.

Примеры

road.in	road.out
2 ABA	1 A
2 ABBAA	2 A B

Задача H. Звёзды за кормой...

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

В последнее время Дятел очень заинтересовался одной игрой. Вкратце правила таковы:

Имеется море размером $(2 \cdot n+1) \times (2 \cdot n+1)$ клеток. Изначально в некоторых клетках находятся острова, в некоторых — вражеские корабли, в центре стоит корабль игрока. За один ход игрок может либо переместиться, либо выстрелить; пропустить ход нельзя.

Перемещение производится на одну из восьми соседних клеток, при условии, что она существует и свободна. При этом считается, что корабль сначала поворачивается на месте носом к выбранной клетке, а затем двигается в неё.

При выстреле корабль производит залп обоими бортами. Ядра при этом летят перпендикулярно текущему курсу корабля. Ядра каждого борта поражают первую встретившуюся в данном направлении цель (корабль или остров), но не далее трёх клеток от корабля игрока (не считая клетку, на которой находится сам корабль игрока). Например, если корабль приплыл из клетки (2, 1) в клетку (1, 2), а затем выстрелил, то он уничтожит вражеский корабль на клетке (4, 5), но не поразит корабль на (5, 6). Если к тому же на клетке (2, 3) будет стоять корабль или остров, то корабль на (4, 5) останется цел.

Изначально корабль игрока стоит на клетке (n, n) с таким направлением, будто он приплыл из клетки $(n-1, n)$.

После каждого хода игрока вражеские корабли одновременно делают ход. Каждый из кораблей ходит на ту из восьми соседних клеток, сумма модулей разностей координат которой и координат корабля игрока минимальна. Формально, если корабль игрока находится на клетке (x_s, y_s), то выбирается такая клетка (x, y), для которой $|x_s-x|+|y_s-y|$ минимально. Если при этом оказывается, что в данной клетке находится остров, то вражеский корабль погибает. Если хотя бы один вражеский корабль попадает на клетку с кораблем игрока, то игрок проигрывает. Если хотя бы два вражеских корабля оказываются на одной клетке, то они погибают и на этой клетке образуются обломки, которые далее действуют так же, как остров, за тем исключением, что снаряды перелетают через обломки, а не поражают их. При уничтожении корабля ядром тоже образуются обломки. На месте острова обломки не образуются.

Игрок побеждает, если на поле нет ни одного живого вражеского корабля.

Дятлу стало очень интересно, можно ли в каждой конкретной ситуации выиграть или нет. Для этого он решил написать программу. Нет, вам не надо ему помогать. Он с этой задачей уже справился, а вот справитесь ли вы?

Формат входного файла

В первой строке входного файла задано число n ($1 \leqslant n \leqslant 6$), определяющее размер поля. Далее следуют $2 \cdot n+1$ строк по $2 \cdot n+1$ символов в каждой. При этом i-й символ $(j+1)$-й строки описывает клетку поля с координатами $(i-1, j-1)$ (клетки нумеруются с 0). Значение символов следующее:

- '`.`' (точка) — пустая клетка.
- '`t`' (t маленькое латинское) — вражеский корабль.
- '`O`' (о большое латинское) — остров.
- '`+`' (плюс) — корабль игрока.

Символ '`+`' всегда присутствует и располагается только в клетке с координатами (n, n). Количество остальных символов может быть любым и ограничено только размерами доски.

Формат выходного файла

В первой строке файла выведите количество ходов, необходимых для выигрыша. В последующих строках выведите координаты корабля после каждого из ходов, по одной паре в строке. Минимизи-

ровать число ходов не обязательно; тем не менее, если вражеских кораблей уже не осталось, дальнейшие ходы делать не следует. Если решения не существует, выведите 'IMPOSSIBLE' в единственной строке файла. Если существует несколько решений, выведите любое.

Примеры

seastars.in	seastars.out
5 ....t ....t... O......t... t......... ....tO ..t..+..... ....O.....Ot ...O....... ....t...... ..t..O.....	IMPOSSIBLE
5 ....t..O.O. ....t...... ....O..+..... .... .t........ .....O.....	7 5 5 5 5 5 5 5 4 5 3 6 2 5 1

Задача I. Суффиксы

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

Назовём *строкой* последовательность из маленьких букв латинского алфавита. Строкой, например, является пустая последовательность "", слово "`aabaf`" или бесконечная последовательность букв "`a`".

i-ый суффикс S_i строки S — это строка S, из которой вырезаны первые i букв; так, для строки $S =$ "`aabaf`" суффиксы будут такими:

$$\begin{aligned} S_0 &= \text{“}\texttt{aabaf}\text{”} \\ S_1 &= \text{“}\texttt{abaf}\text{”} \\ S_2 &= \text{“}\texttt{baf}\text{”} \\ S_3 &= \text{“}\texttt{af}\text{”} \\ S_4 &= \text{“}\texttt{f}\text{”} \\ S_5 = S_6 = S_7 = \ldots &= \text{“”} \end{aligned}$$

Суффиксы определены для всех $i \geqslant 0$.

Циклическое расширение $S^\star$ конечной строки S — это строка, полученная приписыванием её к самой себе бесконечное количество раз. Так,

$$\begin{aligned} S^\star = S_0^\star &= \text{“}\texttt{aabafaabafaa...}\text{”} \\ S_1^\star &= \text{“}\texttt{abafabafabaf...}\text{”} \\ S_2^\star &= \text{“}\texttt{bafbafbafbaf...}\text{”} \\ S_3^\star &= \text{“}\texttt{afafafafafaf...}\text{”} \\ S_4^\star &= \text{“}\texttt{fffffffffff...}\text{”} \\ S_5^\star = S_6^\star = S_7^\star = \ldots &= \text{“”} \end{aligned}$$

По данной строке S выясните, сколько её суффиксов S_i имеют такое же циклическое расширение, как и сама строка S, то есть количество таких i, что $S^\star = S_i^\star$.

Формат входного файла

В первой и единственной строке входного файла задана строка S, состоящая из не менее, чем одной, и не более, чем 100 000 маленьких латинских букв `a-z`.

Формат выходного файла

Выведите в первую строку выходного файла одно число — количество суффиксов строки S, имеющих такое же циклическое расширение, как и она сама.

Примеры

suffixes.in	suffixes.out
aa	2
ab	1
qqqq	4
xyzzyxy	1

Задача J. Доказательство теоремы

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

Преподаватель читает курс лекций, в рамках которого обычно доказывается N различных теорем. Некоторые теоремы могут ссылаться в доказательстве друг на друга. Более точно, каждая теорема $\mathcal{T}_i$ зависит от некоторого набора из C_i других теорем; доказать её можно, лишь доказав любые $\lceil C_i/2 \rceil$ или более из них. При этом структура курса такова, что нет такой теоремы, от которой зависели бы две или более различных теоремы, а также нет цепочки теорем $(\mathcal{T}_{i_1}, \mathcal{T}_{i_2}, \ldots, \mathcal{T}_{i_s})$ такой, что $\mathcal{T}_{i_1}$ зависит от $\mathcal{T}_{i_2}$, $\mathcal{T}_{i_2}$ зависит от $\mathcal{T}_{i_3}$, $\ldots$, $\mathcal{T}_{i_{s-1}}$ зависит от $\mathcal{T}_{i_s}$, а $\mathcal{T}_{i_s}$ — от $\mathcal{T}_{i_1}$.

Однако в этом семестре в связи с обилием праздников, перекрывающихся с лекциями, может не удаться доказать все теоремы курса. Тем не менее, нужно доказать основную теорему курса — это центральный результат всей теории, и именно его, скорее всего, придётся применять слушателям в других курсах в следующем семестре. Поэтому преподаватель хочет расположить теоремы в таком порядке, чтобы основную теорему курса удалось доказать как можно раньше. Затем, если останется время, он сможет вернуться к доказательству других, менее важных, теорем.

Для простоты будем считать, что все теоремы доказываются за одинаковое время. Нужно доказать такое множество теорем и в таком порядке, чтобы основная теорема оказалась доказанной и чтобы общее время доказательства было минимально.

Формат входного файла

В первой строке входного файла записано число N ($1 \leqslant N \leqslant 10\,000$) — количество теорем. Каждая из следующих N строк описывает теоремы, от которых зависит $\mathcal{T}_{i-1}$, где i — номер этой строки во входном файле. Эти строки имеют вид $A_{i,1}\ A_{i,2}\ \ldots A_{i,C_i}\ 0$; здесь $A_{i,j}$ — номер теоремы, от которой зависит $\mathcal{T}_{i-1}$. Среди всех чисел $A_{i,j}$ во входном файле нет двух одинаковых. Основная теорема имеет номер 1. Все числа во входном файле целые.

Формат выходного файла

В первой строке выходного файла выведите K — минимальное количество теорем, которые потребуется доказать. В последующих K строках выведите номера этих теорем в порядке их доказательства, по одному числу в каждой. Если ответов с максимальным K несколько, можно вывести любой из них.

Примеры

theorem.in	theorem.out
2 2 0 0	2 2 1
6 2 3 6 0 4 0 0 0 0 5 0	4 4 3 2 1
3 0 1 0 2 0	1 1