Задача 1. Произведение

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дан набор чисел. Определите, равно ли какое-то из чисел произведению всех остальных.

Формат входных данных

В первой строке входного файла дано целое число N — количество чисел ($2 \leqslant N \leqslant 10^5$). В следующей строке через пробел задано N целых чисел, составляющих исходный набор. Каждое число по абсолютной величине не превосходит 10^9.

Формат выходных данных

В первую строку выходного файла нужно вывести `Yes` или `No`, в зависимости от того, равно ли некоторое число из набора произведению всех остальных чисел или нет. Если ответ `Yes`, то во вторую строку нужно вывести такое число. Если несколько чисел удовлетворяют условию задачи, то можно вывести любое из них.

Примеры

input.txt	output.txt
3 2 4 2	Yes 4
2 2 3	No

Задача 2. Управление «Антилопой»

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Адам Казимирович несется со скоростью v на своём автомобиле «Антилопа гну» по огромной кафедральной площади. Внезапно прямо на его пути на расстоянии d возникает ксёндз, пытающийся охмурить Адама.

Если представить движение машины на плане, то ксёндза можно считать отрезком длины w, перпендикулярным траектории машины, центр которого лежит на этой траектории.

Тут в дело вмешивается Остап — он берет управление в свои руки и хочет избежать столкновения «Антилопы» и ксёндза. Для этого он может сообщать некоторое ускорение автомобилю в каждый момент времени в любом направлении. Причём, в разные моменты времени может отличаться как модуль ускорения, так и его направление. Но, конечно, и Остап не всемогущ, поэтому модуль ускорения в каждый момент времени не должен превышать значения a.

Требуется найти минимально возможное значение a, позволяющее избежать столкновения.

Формат входных данных

В первой строке входного файла задано целое число T — количество тестовых случаев ($1 \leqslant T \leqslant 5 \cdot 10^4$).

В следующих T строках записано по три целых числа v, d и w — начальная скорость автомобиля, расстояние от «Антилопы» до ксёндза и длина отрезка ($1 \leqslant v, d, w \leqslant 1000$).

Формат выходных данных

В выходной файл необходимо вывести T строк, в i-й строке — ответ для i-го тестового случая. Ответом является минимальное возможное значение a, при котором удастся избежать столкновения.

Абсолютная или относительная погрешность каждого ответа не должна превышать 10^{-10}.

Пример

input.txt	output.txt
1 10 3 1	10.510002209123385

Задача 3. Сумасшедший сапёр

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В данной интерактивной задаче вам предлагается сыграть в известную игру «Сапёр».

Действия происходят на прямоугольном клеточном поле размера $H \times W$. На клетках данного поля случайным образом расставлено K мин, их расположение неизвестно игроку. Цель игрока состоит в том, чтобы выяснить, в каких клетках расположены мины.

Изначально все клетки поля считаются «закрытыми». Игрок может «открыть» любую клетку поля. Если в открываемой клетке расположена мина, то игрок проигрывает. В противном случае игроку сообщается количество мин в соседних клетках, и игра продолжается. Две клетки считаются соседними, если они соприкасаются стороной или углом. Игра заканчивается победой, когда на поле остаются закрытыми ровно K клеток — клетки с минами.

Часто в игре возникают ситуации, когда по доступной игроку информации невозможно однозначным образом определить, где расставлены мины. В таких случаях приходится угадывать и пытаться открыть клетку, на которой, возможно, расположена мина. Чтобы минимизировать влияние на игру данных неприятных событий, вам разрешается до шести раз ошибиться, открыв клетку с миной. При открытии такой клетки вам будет сообщено, что здесь расположена мина, клетка не будет считаться открытой, и игра продолжится. Если же вы в седьмой раз откроете клетку с миной, то игра будет проиграна.

Каждое игровое поле сгенерировано следующим образом. Жюри задаёт размеры поля W и H, количество мин K и магическое число S. Затем генерируется псевдослучайная последовательность целых чисел:

$$T_0 = S$$

$$T_{i+1} = (48\,271 * T_i) \bmod 2\,147\,483\,647$$

Затем по этой последовательности генерируются положения мин. Для этого перебираются по порядку элементы T_1, T_2, T_3, ... Для каждого элемента T_i вычисляется номер строки r_i и номер столбца c_i по формулам:

$$r_i = \left\lfloor \frac{T_i}{W} \right\rfloor \bmod H \qquad c_i = T_i \bmod W,$$

и в клетку с этими номерами ставится мина, если её там ещё нет. Процесс расстановки заканчивается, как только количество мин на поле достигает K.

Нумерация строк и столбцов начинается с нуля.

Протокол взаимодействия

В данной задаче вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

При старте вашей программе на стандартный поток ввода подаётся строка, содержащая три целых числа H, W и K. Во всех тестах $H = 16$, $W = 30$, а число K выбрано случайным образом из диапазона от 100 до 200 включительно. Магическое число S не задаётся во входном файле, но жюри выбирает его случайным образом в диапазоне от 100 до 10 000 включительно.

Далее ваша программа должна отправлять запросы открытия на стандартный поток вывода. Каждый запрос должен состоять из одной строки, в которой записаны два числа: номер строки r и номер столбца c открываемой клетки ($0 \leqslant r \leqslant H-1$, $0 \leqslant c \leqslant W-1$).

После каждого отправленного запроса может прийти один из следующих ответов:

- `Empty`, за которым через пробел следует целое число от 0 до 8 — означает, что вы открыли пустую клетку и вам сообщено количество мин в соседних клетках.
- `Boom` — означает, что вы попытались открыть клетку с миной и потратили одну из возможностей ошибиться.
- `Lose` — означает, что вы попытались открыть клетку с миной и проиграли.
- `Win` — означает победу. Ура!

Ответы `Empty` и `Boom` означают, что игра ещё продолжается. После ответов `Lose` или `Win` ваша программа должна завершиться и ничего больше не выводить в стандартный поток вывода. Разрешается открывать одну клетку несколько раз. Количество запросов не должно превышать $2 \cdot W \cdot H$, иначе решение получит вердикт Wrong Answer.

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждой выведенной команды. Иначе решение может получить вердикт Timeout.

Пример

стандартный поток ввода	стандартный поток вывода
16 30 137 Boom Empty 2 Empty 0 Empty 2 Boom Empty 6 ... Win	0 0 0 2 15 29 14 27 15 26 4 15 ... 15 28

Пояснение к примеру

Полностью раскрытое поле в примере (он же первый тест) выглядит так:

```
X2223X20013X310011100123211XXX
24XX3X2001XXX2122X2111XXX22343
X4X32111235X43X5X43X34443X23X3
X3110113XX3X34XXXX33XXX334X5XX
110112X4X4322XX654X334XX2XXX5X
0001X22X23X323XX2X4X212222334X
1112211112XX1234323X20122101X3
X33X2221135533X3X222112XX2012X
2XX33XX22XXXX3X313X201X6X42121
234X445X335X442213X2013XXX2X10
X22X4XX22X33X3X11X22124X633232
X3325X4112X34X312221X3XX3X12XX
3X2X4X41124X4X323X113X534333X3
X2214XX22X3X312XX3203X5X4XX211
11014X53X22121223X102X4X4X4200
0001XX3X21001X101110112122X100
```

Задача 4. Ошибка в коде

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Начинающий программист написал программу на языке C++.

1. Сначала программа считывает матрицу смежности неориентированного графа с N вершинами и записывает её в массив 32-битных целых чисел g. После чтения значение $g[u][v]$ равно единице, если в графе есть ребро между вершинами u и v, и равно 9999, если ребра нет (для $1 \leqslant u \neq v \leqslant N$). На главной диагонали в матрице записаны нули.
2. После считывания матрицы в программе запускается следующий код:

```
for (int w = 1; w < N; w = w + 1) {
    for (int u = 1; u < N; u = u + 1) {
        for (int v = 1; v < N; v = v + 1) {
            g[u][v] = min(g[u][v], g[u][w] + g[w][v]);
        }
    }
}
```

3. После этого полученная матрица записывается в файл.

Программист запустил свою программу на наборе графов и после долгого ожидания решил посмотреть результаты, которые, конечно же, его огорчили. Как вы догадались, программист хотел посчитать матрицу кратчайших расстояний между всеми вершинами в графе. Вот только программа была написана с ошибками. Исправить ошибки программист может и сам, но ждать, когда программа пересчитает результаты, очень долго. Вам же предлагаем написать программу, принимающую на вход полученный неудачливым программистом результат, и возвращающую матрицу кратчайших расстояний.

Формат входных данных

В первой строке входного файла дано целое число N ($1 \leqslant N \leqslant 2\,000$). В i-й из следующих N строк дана последовательность из N чисел, j-е из которых равно значению $g[i][j]$ после выполнения данного выше алгоритма ($0 \leqslant g[i][j] \leqslant 9999$).

Формат выходных данных

В выходной файл необходимо вывести N строк. В i-й из них должна быть последовательность чисел, разделённых пробелами, j-е из которых равно длине кратчайшего пути в графе между вершинами i и j или числу 9999, если пути между этими вершинами нет.

Пример

input.txt	output.txt
3 0 1 1 1 0 9999 1 9999 0	0 1 1 1 0 2 1 2 0

Жюри настоятельно рекомендует использовать в решениях на языке C++ функции **scanf и printf** для ввода/вывода и отправлять решения на компиляторе **Visual C++**. В решениях на Java используйте общепринятый быстрый ввод/вывод. В противном случае ваше решение, скорее всего, **не** уложится в ограничение по времени.

Задача 5. День рождения

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Мальвина хочет подарить на день рождения Буратино неориентированный граф G с множеством вершин V и с множеством ребер E. Буратино исполняется k лет. Мальвина хочет отобразить эту знаменательную для него дату в графе, разбив его на k частей, то есть представив множество вершин графа V в виде k попарно непересекающихся подмножеств V_1, V_2, …, V_k. Разумеется, в этом разбиении все подмножества V_i должны быть непустыми.

А чтобы показать связь между прошедшими k годами, Мальвина хочет разбить граф таким образом, чтобы у каждого ребра (u, v) концевые вершины u и v были либо из одного подмножества, либо из двух соседних подмножеств. Соседними считаются подмножества V_i и V_{i+1} для $i = 1, 2, \ldots k-1$.

Формат входных данных

В первой строке входного файла через пробел записано два целых числа N и k — количество вершин в графе и количество частей, на которые требуется разбить граф ($1 \leqslant N \leqslant 2\,000$, $1 \leqslant k \leqslant 2\,000$).

Граф задан матрицей смежности G. Следующие N строк файла содержат по N символов каждая. j-й символ i-й строки равен `1`, если между вершинами i и j есть ребро и `0`, если между данными вершинами нет ребра. Гарантируется, что матрица симметрична ($g_{i,j} = g_{j,i}$) и что на диагонали матрицы стоят нули ($g_{i,i} = 0$).

Формат выходных данных

В первую строку выходного файла требуется вывести `Yep`, если граф возможно разбить на k частей требуемым образом. Затем требуется вывести k строк. i-ая строка ($i = 1, 2, \ldots, k$) содержит описание i-ой части: сначала требуется вывести количество вершин в V_i, затем через пробел вывести номера входящих в V_i вершин в той же нумерации, которая используется во входных данных. Если существует несколько вариантов разбить граф, то можно вывести любой из них.

Если же граф требуемым образом разбить на k частей нельзя, то требуется вывести `Nope`.

Примеры

input.txt	output.txt
4 3 0110 1010 1101 0010	Yep 1 2 2 1 3 1 4
3 3 011 101 110	Nope

Задача 6. Штрафные баллы за топливо

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Паша участвует в первенстве по ориентированию в пустыне «Величайший Сенсей Ориентирования 2018» (ВСО 2018). Пустыня может быть представлена бесконечной плоскостью. Пустыня заранее подготовлена к соревнованиям — в ней размещено n контрольных точек, пронумерованных последовательно числами от 1 до n, i-я контрольная точка имеет координаты (x_i, y_i).

Первенство проходит по следующим правилам:

1. Всех участников отправляют в начальную контрольную точку с номером s ($1 \leqslant s \leqslant n$).

2. Каждому участнику выдают автомобиль с ёмкостью топливного бака v и расходом топлива, равным одному литру на единицу длины, то есть на прохождение любого расстояния тратится количество литров топлива, равное этому расстоянию. Изначально топливный бак пуст.

3. Пополнять топливный бак участники могут только на контрольных точках, для каждой из которых известно количество штрафных баллов p_i, начисляемых за приобретение одного литра топлива в этой контрольной точке. Разрешается пополнять бак на нецелое количество литров.

4. Участникам сообщают номер конечной контрольной точки f ($1 \leqslant f \leqslant n$, $f \neq s$). Цель участников — добраться из начальной контрольной точки s в конечную контрольную точку f, набрав минимальное количество штрафных баллов.

Паша не знает, какие именно контрольные точки s и f выберут организаторы, поэтому ему важно заранее знать математическое ожидание количества полученных штрафных баллов. Конечно же, Паша, как настоящий профессионал, всегда выбирает маршрут, при котором ему будет начислено наименьшее количество штрафных баллов. Все допустимые варианты выбора контрольных точек s и f можно считать равновероятными.

Напишите программу, которая поможет Паше определить искомое математическое ожидание.

Формат входных данных

Первая строка входного файла содержит два целых числа n и v — количество контрольных точек и ёмкость топливного бака соответственно ($2 \leqslant n \leqslant 200$, $1 \leqslant v \leqslant 10^5$).

Далее следует n строк, каждая из которых описывает очередную контрольную точку тремя целыми числами x_i, y_i и p_i — координаты этой контрольной точки и количество штрафных баллов за литр топлива в ней соответственно ($-10^4 \leqslant x_i, y_i \leqslant 10^4$, $1 \leqslant p_i \leqslant 10^6$). Никакие две контрольные точки не совпадают.

Гарантируется, что ёмкости топливного бака достаточно, чтобы можно было из любой контрольной точки добраться до любой другой, возможно дозаправляясь в пути.

Формат выходных данных

В выходной файл необходимо вывести одно вещественное число — искомое математическое ожидание количества штрафных баллов. Ответ будет считаться правильным, если его абсолютная или относительная погрешность не превышает 10^{-6}.

Примеры

input.txt	output.txt
5 3 0 0 3 0 2 2 1 1 5 3 1 2 2 2 4	6.634062043
3 100000 0 0 1000 1 1 1 2 0 1000	943.751850623645282

Пояснение к примеру

В первом примере всего 20 вариантов выбора начальной и конечной контрольных точек. Во всех вариантах, кроме четырёх, оптимальный маршрут заключается в том, чтобы купить нужный объём бензина на первой контрольной точке и ехать напрямую на конечную контрольную точку.

Если ехать из (0, 2) в (3, 1) или обратно, то напрямую проехать нельзя из-за недостаточной ёмкости бака. Наилучшим вариантом будет заправиться на максимум в начальной контрольной точке, потом проехать через точку (2, 2), дозаправившись там ещё немного. Получается общий штраф: $2 \cdot 3 + 4 \cdot (\sqrt{2} - 1)$.

Аналогично, при выезде из $(0, 0)$ в $(3, 1)$ и обратно оптимальным решением будет проехать через контрольную точку (1, 1) для дозаправки.

Во втором примере в некоторых случаях выгодно ехать через промежуточную точку, чтобы заправиться.

Задача 7. Проверка уровня

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	6 секунд
Ограничение по памяти:	256 мегабайт

Вася всё ещё любит играть в компьютерные игры, и по-прежнему работает тестировщиком игр. С момента прошлой встречи изменилось немногое: он уже задумывается о том, как бы ему научиться программировать, но пока не сделал решительного шага. В результате, он по-прежнему сидит в комнате два-на-два-на-два и тестирует всё ту же браузерную игру.

Недавно дизайнеры уровней постановили, что всё-таки можно делать уровни с нелинейным прохождением. Теперь *базовая карта уровня* игры представляется набором из N проходимых клеток на бесконечном клеточном поле. Игрок, оружие и монстры всегда находятся в проходимых клетках. За один шаг игрок может перейти в любую проходимую клетку, смежную по стороне с текущей клеткой.

Увы, эпохальное решение дизайнеров по смене курса доставило Васе одни проблемы. Сейчас он занимается тестированием самого первого уровня игры. Проблема в том, что у игрока в начале игры нет никакого оружия, а без оружия битва с любым монстром обречена на *провал*, бессмысленный и беспощадный. Как тестировщика, Васю заставили проверять, что игрок гарантированно найдёт хоть какое-то оружие до того момента, как он впервые встретит монстра. При этом предполагаемый интеллектуальный уровень целевой аудитории так же низок, как и всегда. Так что, если есть хоть какая-то возможность прийти к провалу, обязательно найдутся игроки, которые это сделают.

К сожалению, дизайнеры относятся к первому уровню с нездоровой долей перфекционизма. Хотя базовая карта этого уровня уже утверждена и зафиксирована, каждый день Васе присылают по несколько вариантов расстановки объектов на ней. *Расстановка объектов* показывает, в каких клетках уровня находятся монстры, в каких клетках уровня находится оружие, и в какой клетке изначально расположен игрок. Васе нужно для каждой расстановки объектов определить, возможен ли для неё бессмысленный провал или нет.

Помогите Васе: напишите программу, которая будет быстро анализировать расстановки объектов, пока Вася гамает во второй PoE. Учтите, что, когда игрок попадает в клетку с оружием, он автоматически его забирает, а когда он попадает в клетку с монстром, то он вынужден биться с этим монстром.

Формат входных данных

В первой строке входного файла задано целое число N — количество проходимых клеток на базовой карте уровня ($1 \leqslant N \leqslant 3 \cdot 10^5$). Следующие N строк описывают эти клетки: в каждой строке два целых числа x и y — её координаты ($0 \leqslant x, y \leqslant 10^6$). Гарантируется, что все эти клетки различные.

В следующей строке записано одно целое число T — количество расстановок объектов, которые требуется проанализировать ($1 \leqslant T \leqslant 10^5$). Далее следует T блоков, каждый блок описывает одну расстановку объектов.

Блок начинается со строки, в которой записано два целых числа: M — количество монстров и W — количество клеток с оружием ($0 \leqslant M, W \leqslant 10^5$). В следующей строке записаны координаты клетки, в которой изначально расположен игрок. В следующих M строках записаны координаты клеток с монстрами, и, наконец, в последних W строках блока записаны координаты клеток, в которых лежит оружие. Гарантируется, что все эти $(M+W+1)$ клетки проходимые, и что все они отличаются друг от друга.

Суммарное количество монстров по всем расстановкам объектов не превышает 10^5. Аналогично, суммарное количество клеток с оружием по всем расстановкам не превышает 10^5.

Формат выходных данных

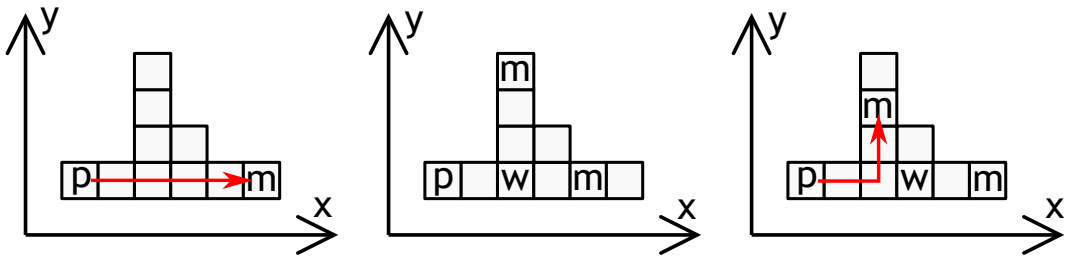
Для каждой расстановки объектов нужно вывести ответ на задачу в отдельную строку выходного файла. Ответы должны быть выведены в том порядке, в котором расстановки объектов заданы во входном файле. Если игрок может закончить игру провалом, то нужно вывести в качестве ответа слово `fail`, а иначе — слово `ok`.

Пример

input.txt	output.txt
10 1 1 2 1 3 1 4 1 5 1 6 1 4 2 3 2 3 3 3 4 3 1 0 1 1 6 1 2 1 1 1 5 1 3 4 3 1 2 1 1 1 3 3 6 1 4 1	fail ok fail

Иллюстрация

Три расстановки объектов из примера (`p` — игрок, `m` — монстр, `w` — оружие):



Задача 8. Граф пересечения

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Иннокентий продолжает писать геометрические алгоритмы, но теперь он перешёл от треугольных сеток к твёрдым телам. Сейчас он хочет написать алгоритм построения графа пересечения двух тел, который необходим в дальнейшем для выполнения, к примеру, булевых операций над телами. Следует заметить, что при построении графа пересечения учитываются только границы тел, сами внутренние области тел никак в задаче не участвуют. Иннокентий знает, что построить граф пересечения для двух произвольных тел в стандартном для САПР-области представлении очень сложно. Поэтому он решил начать с простого: научиться строить граф пересечения для боксов A и B.

Бокс — это прямоугольный параллелепипед со сторонами, параллельными осям координат. Граница бокса состоит из *топологических элементов* трёх типов:

- `V` — вершина, в геометрическом смысле является точкой;
- `E` — ребро, в геометрическом смысле представляется отрезком;
- `F` — грань, в геометрическом смысле представляется прямоугольником;

Всего у каждого бокса 8 вершин, 12 рёбер и 6 граней.

В данной задаче будем считать, что рёбра и грани представляются соответствующими множествами точек в пространстве **без краёв**. То есть ребро является отрезком, не включающим свои концевые точки, а грань является прямоугольником, не включающим свои углы и стороны. Благодаря этому определению получается, что топологические элементы не имеют общих точек, и все вместе составляют в точности границу бокса.

Граф пересечения состоит из взаимосвязных *элементов пересечения*, которые определяются следующим образом. Рассмотрим все возможные пары: топологический элемент u из бокса A и топологический элемент v из бокса B (всего 676 пар). Для каждой такой пары (u, v) найдём их пересечение как пересечение множеств. Если это пересечение непустое, то оно определяет элемент пересечения, который получается одного из трёх типов:

- `p` — точка: одна точка;
- `c` — кривая: множество точек, составляющее отрезок;
- `s` — поверхность: множество точек, составляющее прямоугольник;

Граф пересечения состоит из всех элементов пересечения, полученных таким образом.

Элементы пересечения связаны друг с другом. Для каждой кривой пересечения сохраняются две точки пересечения, которые находятся на её концах. Для каждой точки пересечения сохраняется список кривых пересечения, инцидентных ей. Считается, что любая кривая пересечения инцидентна своим концевым точкам, и не инцидентна никаким другим точкам пересечения. Для поверхности пересечения сохраняется множество кривых пересечения, которые находятся на её краю, и таким образом ограничивают её.

Формат входных данных

В первой строке входного файла задано целое число T — количество тестовых случаев ($1 \leqslant T \leqslant 5\,000$). Далее следует T блоков.

Каждый блок состоит из двух строк: в первой строке задан бокс A, а во второй — бокс B. Бокс описывается шестью целыми числами x_1, y_1, z_1, x_2, y_2, z_2, по абсолютной величине не превышающими 10^3. Числа x_1, y_1, z_1 задают координаты «минимальной» вершины бокса, а числа x_2, y_2, z_2 задают координаты «максимальной» вершины ($x_1 < x_2$, $y_1 < y_2$, $z_1 < z_2$).

Формат выходных данных

В выходной файл нужно вывести T графов пересечения — ответы на тестовые случаи из входного файла. После каждого графа пересечения нужно вывести в отдельной строке `===` (три символа равенства). Все числа в выходном файле должны быть целыми.

В описании графа пересечения очень много связей между элементами пересечения и топологическими элементами. Для того, чтобы можно было их описать, вам требуется самостоятельно пронумеровать все эти элементы. Все топологические элементы бокса A нужно пронумеровать целыми числами от 1 до 26 в любом порядке. Аналогично, все топологические элементы бокса B нужно пронумеровать от 1 до 26 в любом порядке. Кроме того, все элементы пересечения нужно пронумеровать от 1 до K в том порядке, в котором вы выводите их в выходной файл, где K — общее количество элементов пересечения.

В первую строку описания графа пересечения нужно вывести три числа: P — количество точек пересечения, C — количество кривых пересечения и S — количество поверхностей пересечения (разумеется, $P + C + S = K$). Далее должны быть описаны по порядку: P точек пересечения, C кривых пересечения и S поверхностей пересечения (по одному элементу пересечения в строке).

Описание каждого элемента пересечения должно начинаться с его номера, после чего должен стоят трёхбуквенный код элемента пересечения, составленный по правилу:

1. Первая буква — тип элемента пересечения: точка `p`, кривая `c` или поверхность `s`.
2. Вторая буква — тип топологического элемента u: вершина `V`, ребро `E` или грань `F`.
3. Третья буква — тип топологического элемента v.

Здесь u и v — топологические элементы из боксов A и B, соответственно, при пересечении которых получился этот элемент пересечения. Далее нужно вывести два целых числа: номера этих топологических элементов u и v в этом порядке. Наконец, в зависимости от типа элемента пересечения нужно вывести ещё дополнительную информацию.

Для точки пересечения нужно дополнительно записать три её координаты, а также количество инцидентных кривых пересечения и список их номеров. Номера инцидентных кривых можно выводить в любом порядке.

Для кривой пересечения нужно дополнительно записать номера двух точек пересечения, которые находятся на её концах. Можно выводить эти два номера в любом порядке.

Для поверхности пересечения нужно дополнительно записать количество кривых пересечения на её краю и список их номеров. Номера кривых на краю можно выводить в любом порядке.

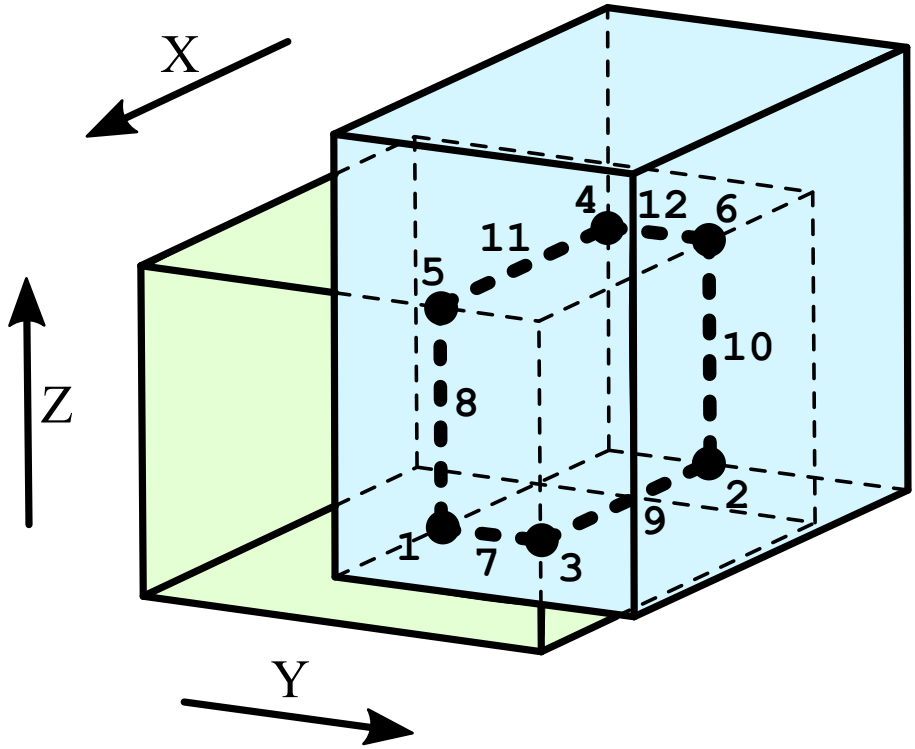
Не обязательно отделять числа в файле друг от друга ровно одним пробелом: можно разделять их несколькими пробелами, как это сделано в примере.

Пример

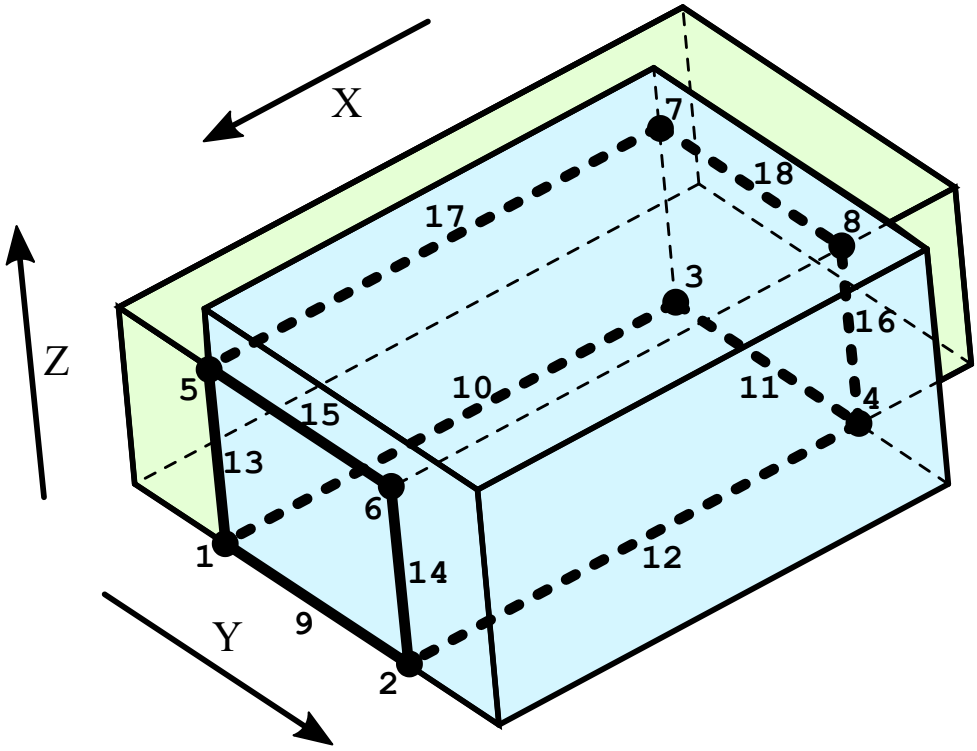
input.txt	output.txt
2 0 0 0 5 4 3 2 3 1 7 6 5 0 2 6 10 8 9 2 4 6 10 10 10	6 6 0 1 pFE 14 2 5 3 1 2 7 8 2 pFE 16 4 2 4 1 2 9 10 3 pEF 17 5 5 4 1 2 7 9 4 pFE 22 10 2 3 3 2 11 12 5 pEF 23 11 5 3 3 2 8 11 6 pEF 25 13 2 4 3 2 10 12 7 cFF 14 5 1 3 8 cFF 14 11 1 5 9 cFF 16 5 2 3 10 cFF 16 13 2 6 11 cFF 22 11 4 5 12 cFF 22 13 4 6 === 8 10 2 1 pEV 6 3 10 4 6 3 9 10 13 2 pVE 9 6 10 8 6 3 9 12 14 3 pFV 5 1 2 4 6 2 10 11 4 pEE 8 4 2 8 6 3 11 12 16 5 pEE 23 12 10 4 9 3 13 15 17 6 pVF 26 14 10 8 9 2 14 15 7 pFE 22 10 2 4 9 2 17 18 8 pEF 25 13 2 8 9 2 16 18 9 cEE 6 6 1 2 10 cFE 5 2 1 3 11 cFE 5 4 3 4 12 cEF 8 5 2 4 13 cFE 14 12 1 5 14 cEF 17 14 2 6 15 cEF 23 14 5 6 16 cFF 16 13 4 8 17 cFF 22 11 5 7 18 cFF 22 13 7 8 19 sFF 5 5 4 9 10 11 12 20 sFF 14 14 4 9 13 14 15 ===

Иллюстрация

В первом тестовом случае боксы протыкают друг друга углом. Граф пересечения показан на картинке жирным. Он состоит из шести точек и шести кривых, которые заслонены от наблюдателя боксом B. У элементов пересечения подписаны номера.



Во втором тестовом случае у боксов совпадает нижняя и передняя плоскости, в результате чего образуются поверхности пересечения 19 и 20 соответственно (на картинке они **не** нарисованы). Все остальные элементы выделены жирным, и их номера подписаны.



Задача 9. В поисках стула

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Киса и Ося в поисках последнего двенадцатого стула попали на планету Плюк. Им точно известны их координаты и координаты стула и, естественно, они хотят как можно скорее с ним воссоединиться. Да вот незадача — они могут передвигаться всюду по поверхности планеты, за исключением нескольких запретных круглых областей — всех туда попавших заточают в круглую клетку и заставляют петь песню «Мама, мама, что я буду делать». И все бы ничего, в какие только перипетии наши герои не попадали, но вот только подобные задержки в их планы никак не входят — а вдруг их исполнение очень понравится господину ПЖ и он их никогда не выпустит. Поэтому эти области они обходят стороной. Ваша задача — помочь им найти длину кратчайшего пути от места их нынешнего положения до стула, минуя все запретные области.

Формат входных данных

В первой строке входного файла содержатся два числа R и N — радиус планеты и количество запретных круглых областей ($1000 \leqslant R \leqslant 10000, 0 \leqslant N \leqslant 20$).

Далее идут N строк, каждая из которых содержит по три числа: ϕ_i, ψ_i — соответственно широта и долгота центра области и r_i — её радиус ($1 \leqslant r_i \leqslant R/2$). Две последние строки содержат координаты ϕ_s, ψ_s и ϕ_t, ψ_t — соответственно широта и долгота точки старта и точки финиша.

Все числа во входных данных целые. Широта и долгота задаются в градусах. Широта находится в пределах от -90 градусов до 90 градусов, а долгота — в пределах от -180 до 180 градусов.

В запретную область радиуса r входят точки, до которых можно дойти от центра области, пройдя по поверхности планеты путь длиной меньше r.

Гарантируется, что расстояние по поверхности планеты от точки старта до любой точки любой запретной области, а также до точки финиша не превосходит $\frac{3}{2}R$. Запретные области могут пересекаться. Центры запретных областей попарно различны. Гарантируется, что точки старта и финиша не попадают ни в одну запретную область и, более того, минимальное расстояние от этих точек до любой запретной области не менее $R/1000$. Также гарантируется, что при изменении любой характеристики любого пятна не более чем на 10^{-3} взаимное расположение пятен не изменится, то есть если они пересекались, то и останутся пересекающимися и наоборот.

Формат выходных данных

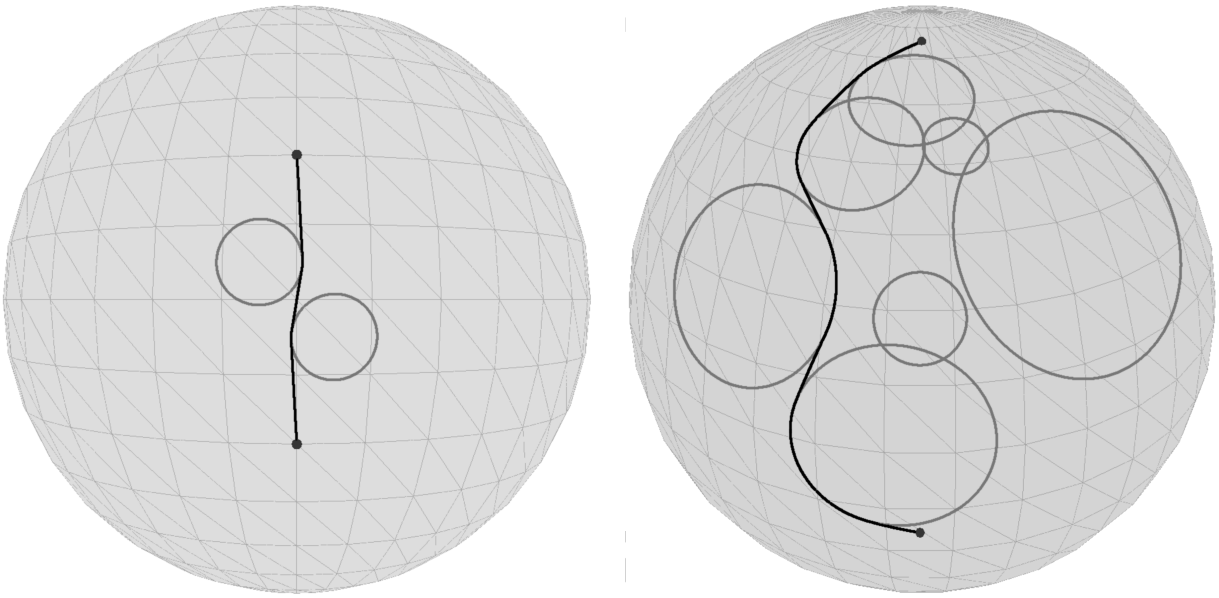
В выходной файл необходимо вывести одно вещественное число — длину кратчайшего пути. Абсолютная или относительная погрешность ответа не должна превышать 10^{-8}. Если пути нет, то вывести `-1`.

Примеры

input.txt	output.txt
1000 2 5 5 100 -5 -5 100 -20 0 20 0	700.830526321397656
1000 7 11 49 253 30 75 253 52 59 158 62 48 157 37 14 348 29 45 107 53 37 79 -5 45 75 45	1715.42276690714903

Иллюстрация

Ниже показано расположение запретных областей и оптимальный путь по поверхности планеты для первого и второго тестов соответственно.



Задача 10. Офис

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Офис компании N* имеет несколько комнат, которые расположены вдоль довольно длинного коридора. В каждой комнате есть несколько рабочих мест. В компании работает много сотрудников, которым периодически приходится бегать в начало и в конец этого коридора: там находятся важные с производственной точки зрения объекты.

В какой-то момент руководство компании решило, что работа будет более эффективной, если сделать размещение сотрудников оптимальным, и наняло консалтинговую фирму, чтобы узнать, как же их рассадить. Консалтинговая фирма сперва измерила длину коридора и узнала, что она равна L, и затем предложила следующий критерий: размещение сотрудников оптимально, если суммарное расстояние, которое они проходят по коридору за день минимально. Если сотрудник сидит в комнате, находящейся на расстоянии p от начала коридора, то, единожды сбегав до его начала, он проходит по коридору расстояние $2p$, а, единожды сбегав до его конца, — расстояние $2(L-p)$. Дальнейший анализ позволил узнать, сколько раз на дню каждый из сотрудников бегает в начало и в конец коридора.

Теперь, когда все данные известны, вам осталось вычислить оптимальное размещение сотрудников. Естественно, число сотрудников, которые могут быть посажены в одну комнату, ограничено числом рабочих мест в ней.

Формат входных данных

В первой строке входного файла записаны три целых числа N, M и L — количество комнат, количество сотрудников и длина коридора ($1 \leqslant N \leqslant 10^5, 1 \leqslant M \leqslant 10^5, 2 \leqslant L \leqslant 10^8$).

Каждая из следующих N строк содержит два целых числа P_i и C_i, характеризующих i-ую комнату — расстояние от начала коридора до комнаты и количество мест для сотрудников в этой комнате($0 < P_i < L$, $1 \leqslant C_i \leqslant 10^5$).

В каждой из следующих M строк записаны два целых числа A_i и B_i, которые характеризуют i-ого сотрудника — сколько раз на дню он бегает из своей комнаты в начало коридора и в его конец соответственно ($0 \leqslant A_i, B_i \leqslant 10^5$).

Суммарное количество мест не превосходит 10^6. Гарантируется, что мест хватает для всех сотрудников.

Формат выходных данных

В первую строку выходного файла необходимо вывести целое число — суммарное расстояние, которое будут проходить сотрудники за день при их оптимальном размещении.

Следующие N строк должны содержать информацию о таком размещении. Первым в i-ой из этих строк должно быть записано целое число S_i — количество сотрудников, рабочие места которых будут находиться в i-ой комнате. Далее в этой же строке должно быть расположено S_i целых чисел — номера этих сотрудников в произвольном порядке.

Если существует несколько решений, выведите любое из них. Сотрудники нумеруются в порядке задания во входном файле, начиная с единицы.

Пример

input.txt	output.txt
4 9 5 1 2 2 3 3 4 4 2 3 0 3 1 3 1 0 2 1 2 1 2 1 1 2 2 3 3	128 2 1 3 2 2 9 3 6 7 8 2 4 5

Задача 11. Рекурсивная схема

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Имеется микросхема, устроенная рекурсивным образом. На плане схемы имеется всего N точек-контактов, некоторые пары контактов соединены напрямую проводящими дорожками. Кроме того, внутри схемы расположено S подсхем, каждая из которых является точной копией рассматриваемой схемы.

Контакты на плане схемы делятся на три типа:

1. Входные контакты схемы (K штук). Это единственные контакты, к которым можно подключать дорожки снаружи рассматриваемой схемы.
2. Входные контакты вложенных подсхем ($S \cdot K$ штук).
3. Вспогомательные контакты.

Все эти контакты могут быть соединены проводящими дорожками друг с другом без каких-либо ограничений.

По проводящим дорожкам может распространяться сигнал. Когда сигнал доходит до контакта, он может далее пойти по любой дорожке, соединённой с этим контактом. Если сигнал подходит снаружи ко входному контакту подсхемы, то он может зайти внутрь подсхемы и далее распространяться по её дорожкам. Если сигнал подходит изнутри ко входному контакту подсхемы, то он может выйти из подсхемы и далее распространяться снаружи неё (если там есть дорожки и она сама является чьей-то подсхемой).

Расмотрим самую внешнюю схему, снаружи которой ничего нет. Требуется определить, соединены ли два заданных входных контакта этой схемы дорожками. Контакты соединены, если сигнал может дойти от одного контакта до другого, возможно заходя по пути внутрь различных подсхем конечное количество раз.

Кроме самого факта соединения, требуется также определить, насколько глубоко сигналу нужно заходить внутрь подсхем, чтобы дойти от одного контакта до другого. Внешняя схема имеет уровень вложенности 0, её подсхемы умеют уровень вложенности 1, подсхемы этих подсхем имеют уровень вложенности 2, и так далее до бесконечности. Для произвольного пути, по которому идёт сигнал, можно определить критическую глубину — максимальный уровень вложенности среди всех схем, дорожки которых участвуют в этом пути. Требуется определить минимальное значение критической глубины для пути между заданными входными контактами внешней схемы.

Формат входных данных

В первой строке записано четыре целых числа: N — количество контактов на плане схемы, K — количество входных контактов у схемы, S — количество подсхем в схеме, M — количество проводящих дорожек на плане схемы ($1 \leqslant K \leqslant 100\,000$, $0 \leqslant S \leqslant 1\,000$, $K(S+1) \leqslant N \leqslant 100\,000$, $0 \leqslant M \leqslant 100\,000$).

В следующих M строках заданы проводящие дорожки на плане схемы. Каждая дорожка задаётся двумя целыми числами a и b — номерами контактов на плане, которые напрямую соединяет дорожка ($1 \leqslant a \neq b \leqslant N$).

Контакты на плане схемы нумеруются по порядку числами от 1 до N. Входные контакты схемы имеют номера от 1 до K. Входные контакты t-ой подсхемы имеют номера от $tK+1$ до $tK+K$ (для $1 \leqslant t \leqslant S$). При этом j-ый входной контакт на плане t-ой подсхемы является $(tK+j)$-ым контактом на плане внешней схемы. Оставшиеся контакты (если есть), являются

вспомогательными.

В следующей строке записано целое число Q — количество запросов ($1 \leqslant Q \leqslant 100\,000$). В каждой из оставшихся Q строк записан один запрос, на который необходимо найти ответ. Каждый запрос задаётся двумя целыми числами u и v — номерами входных контактов внешней схемы ($1 \leqslant u \neq v \leqslant K$).

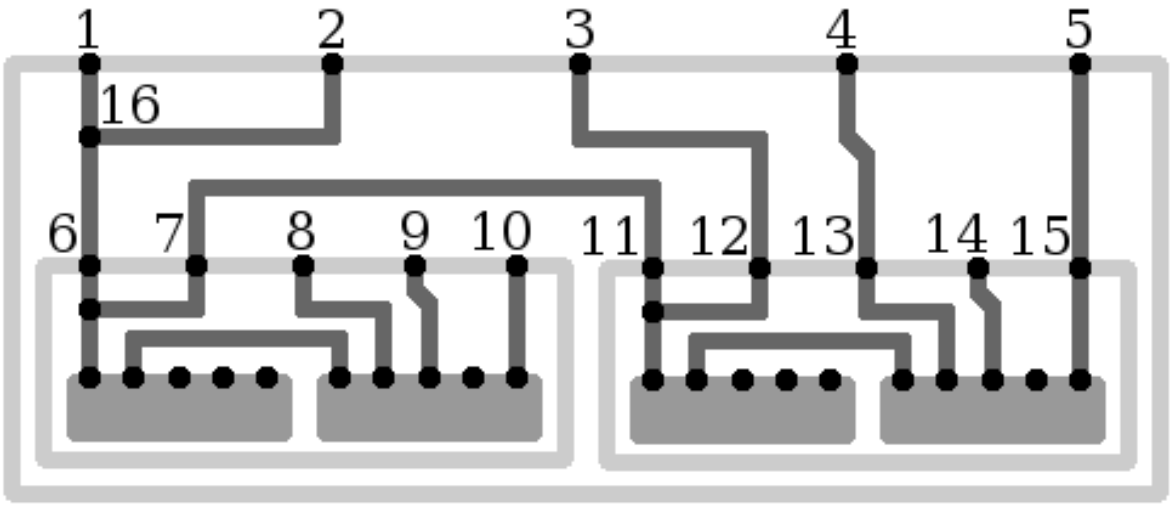
Формат выходных данных

В выходной файл нужно вывести Q целых чисел, по одному числу в строке. Каждое r-ое число должно быть ответом на r-ый запрос: на какую глубину вложенности необходимо опускаться, чтобы добраться от одного входного контакта до другого. Если добраться от одного контакта до другого нельзя, то нужно вывести число -1 вместо глубины.

Пример

input.txt	output.txt
16 5 2 7 1 16 2 16 6 16 7 11 3 12 4 13 5 15 4 1 2 2 3 3 4 4 5	0 1 2 -1

Иллюстрация



Задача 12. Подпоследовательности

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
	8 секунд (для Java)
Ограничение по памяти:	512 мегабайт

В НИИ Данных Строк строку s называют *хорошей*, если у неё количество различных подпоследовательностей чётно. Строку t называют подпоследовательностью строки s, если её можно получить из s вычёркиванием некоторого набора символов. Сама строка s и пустая строка также считаются подпоследовательностями s. Всего в строке длины l имеется 2^l подпоследовательностей, однако некоторые из них могут оказаться одинаковыми. Например, у трёхбуквенной строки abb только 6 различных подпоследовательностей: пустая, a, b, ab, bb, abb.

На столе Исследователя Строк лежала строка s, и он пытался определить, является ли она хорошей. Он, конечно, понимал, что подсчёт количества различных подпоследовательностей — дело нелегкое, и вместо того, чтобы незамедлительно к этому делу приступить, он отправился пить чай с булочками.

По возвращении Исследователь Строк обнаружил, что кто-то сделал $N-1$ разрез в строке s, и она распалась на N непустых подстрок $s_1, s_2, \ldots, s_N$, разбросанных на столе. Исходную строку s исследователь не помнит. Тем не менее, он интересуется, сколько существует перестановок $p = (p_1, \ldots, p_N)$ таких, что склеенная строка $s_{p_1} s_{p_2} \ldots s_{p_N}$ хорошая. Следует заметить, что всего существует $N!$ перестановок, и все эти перестановки считаются различными, даже если какие-то подстроки s_i совпадают.

Формат входных данных

В первой строке входного файла дано целое число N — количество подстрок ($2 \leqslant N \leqslant 20$). В i-й из следующих N строк дана подстрока s_i.

Все строки s_i непустые и состоят лишь из маленьких букв латинского алфавита. Гарантируется, что сумма длин всех строк s_i не превышает 10^5.

Формат выходных данных

В выходной файл необходимо вывести целое число: количество таких перестановок, что конкатенация строк s_i в порядке перестановки является хорошей.

Пример

input.txt	output.txt
3 a a baa	4

Пояснение к примеру

Строка $a + a + baa$ имеет 14 подпоследовательностей, строка $a + baa + a$ имеет 13 подпоследовательностей, а строка $baa + a + a$ имеет 10 подпоследовательностей. При этом каждую из этих строк можно получить с помощью двух перестановок, т.к. подстрока a дана в двух экземплярах.