Технокубок

Технокубок 2017: Условия и разбор задач заключительного этапа

A. Андрюша и носки (максимум 500 баллов)

ограничение по времени на тест 2 секунды

ограничение по памяти на тест 256 мегабайт

ввод стандартный ввод

вывод стандартный вывод

Андрюша очень аккуратный и чистоплотный мальчик, поэтому он всегда следит за порядком в своей комнате. В том числе, он очень внимательно относится к укладыванию своих вещей в шкаф.

Сегодня перед ним встала нелегкая задача — сложить в шкаф носки. У Андрюши есть n разных пар носков, которые изначально сложены в мешок. Пары носков пронумерованы от 1 до n. Андрюша хочет убрать носки в шкаф по парам. Для этого он по одному достает носки из мешка, и для каждого вытащенного носка смотрит, доставал ли он уже его пару. Если нет, что значит, что пара текущего носка все еще находится в мешке, то Андрюша кладет текущий носок на стол перед собой. Иначе он убирает текущий носок и его пару в шкаф.

Андрюша очень внимательный, поэтому он запомнил, в каком порядке доставал носки из мешка. Теперь он задумался, а какое максимальное количество носков одновременно лежало перед ним на столе? Помогите Андрюше ответить на этот вопрос.

Входные данные

В первой строке находится единственное целое число n ($1 \le n \le 10^5$) — число пар носков.

Во второй строке через пробел перечислены $2n$ чисел $x_1, x_2, ..., x_{2n}$ ($1 \le x_i \le n$), которые описывают, в каком порядке Андрюша доставал носки из мешка. А именно, число x_i означает, что i-м по порядку Андрюша достал носок из пары x_i.

Гарантируется, что Андрюша достал ровно два носка из каждой пары.

Выходные данные

В единственной строке выведите одно число — максимальное число носков, которые когда-либо лежали на столе перед Андрюшей.

Примеры

Входные данные

1

1 1

Выходные данные

1

Входные данные

3

2 1 1 3 2 3

Выходные данные

2

Примечание

В первом примере из условия Андрюша достает из мешка носок из первой пары и кладет его на стол. Затем он достает следующий носок, который также из первой пары, так что он сразу же берет оба носка и убирает их в шкаф. Таким образом, максимум один носок лежал на столе.

Рассмотрим, что происходит во втором примере:

- Сначала на столе ничего нет, Андрюша достает носок из пары номер 2.
- На столе лежит носок (2). Андрюша достает носок из пары номер 1 и кладет его на стол.
- На столе лежат носки (1, 2). Андрюша достает носок из пары номер 1, но такой носок уже есть на столе. Поэтому Андрюша убирает оба носка в шкаф.
- На столе лежит носок (2). Андрюша достает носок из пары номер 3 и кладет его на стол.
- На столе лежат носки (2, 3). Андрюша достает носок из пары номер 2, но такой носок уже есть на столе. Поэтому Андрюша убирает оба носка в шкаф.
- На столе лежит носок (3). Андрюша достает носок из пары номер 3, но такой носок уже есть на столе. Поэтому Андрюша убирает оба носка в шкаф.

Таким образом, на столе единовременно лежало максимум два носка.

Разбор задачи:

Простая задача на реализацию. Сохраним в массиве, лежит ли носок каждого из типов на столе, также будем хранить количество носков и наибольшее значение этого количества. Теперь будем обрабатывать носки по одному и обновлять все необходимые значения.

Сложность: $O(n)$ времени и памяти.

B. Место встречи изменить нельзя (максимум 1000 баллов)

ограничение по времени на тест 5 секунд

ограничение по памяти на тест 256 мегабайт

ввод стандартный ввод

вывод стандартный вывод

Главная улица в Байтсити имеет вид длинной прямой с юга на север. Для удобства ориентирования на прямой введены координаты, измеряемые в метрах от самого южного дома в сторону севера.

В некоторых точках улицы сейчас находятся n друзей, причём i-й из них находится в точке с координатой x_i метров и может передвигаться с максимальной скоростью v_i метров в секунду в любом из двух направлений вдоль улицы — на юг или на север.

Перед вами стоит задача определить минимальное время, за которое все n друзей смогут встретиться в одной точке на главной улице. Обратите

внимание, что точка, в которой встретятся друзья, не обязана иметь целочисленную координату.

Входные данные

В первой строке находится единственное целое число n ($2 \le n \le 60\,000$) — количество друзей.

Во второй строке находятся n целых чисел $x_1, x_2, ..., x_n$ ($1 \le x_i \le 10^9$) — текущие координаты друзей в метрах.

В третьей строке находятся n целых чисел $v_1, v_2, ..., v_n$ ($1 \le v_i \le 10^9$) — максимальные скорости друзей в метрах в секунду.

Выходные данные

Выведите минимальное время (в секундах), за которое все n человек смогут встретиться в одной точке.

Ваш ответ будет считаться правильным, если его абсолютная или относительная погрешность не превышает 10^{-6}. Формально, пусть ваш ответ равен a, а ответ жюри — b. Ваш ответ будет считаться правильным, если $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$.

Примеры

Входные данные

3

7 1 3

1 2 1

Выходные данные

2.000000000000

Входные данные

4

5 10 3 2

2 3 2 4

Выходные данные

1.400000000000

Примечание

В первом примере все друзья могут встретиться в точке 5 через 2 секунды. Для этого первый друг должен идти все время со своей максимальной скоростью на юг, а второй и третий должны идти со своими максимальными скоростями на север.

Разбор задачи:

Воспользуемся бинарным поиском по ответу. Внутри бинарного поиска необходимо проверять, можно ли всем друзьям встретиться за t секунд. За это время i-ый друг может оказаться в любой точке отрезка $[x_i - tv_i, x_i + tv_i]$. Чтобы встреча была возможна, какая-то точка должна принадлежать всем этим отрезкам, то есть, их пересечение должно быть непусто.

Удобный способ пересечения набора отрезков $[l_1, r_1], ..., [l_n, r_n]$ заключается в том, чтобы посчитать $L = max\ l_i$ и $R = min\ r_i$. Если $L \le R$, то $[L, R]$ и есть искомое пересечение, иначе пересечение пусто.

Сложность: $O(n \log(\varepsilon^{-1}))$ времени и $O(n)$ памяти. Здесь ε — требуемая относительная погрешность.

C. Андрюша и разноцветные шарики (максимум 1250 баллов)

ограничение по времени на тест 2 секунды

ограничение по памяти на тест 256 мегабайт

ввод стандартный ввод

вывод стандартный вывод

Андрюша каждый день с самого детства ходит через городской парк. Дорожки и полянки парка все время казались ему слишком одинаковыми, и однажды он решил украсить их и сделать разнообразными.

Парк состоит из n полянок, которые соединены между собой (n - 1) двусторонними дорожками, причем от каждой полянки можно дойти по дорожкам до любой другой.

Андрюша решил на каждой полянке повесить один цветной шарик. Цвета шариков задаются целыми положительными числами, начиная с 1. Чтобы парк стал более разнообразным, Андрюша решил выбирать цвета шариков по-особенному. А именно, он хочет развесить шарики так, чтобы для любых трех попарно различных полянок a, b и c таких, что a и b соединены дорожкой, и b и c соединены дорожкой, цвета шариков на этих полянках были попарно различными.

Чтобы не тратить много денег на покупку шариков, Андрюша хочет использовать как можно меньше различных цветов. Так как Андрюша не очень силен в программировании, он просит вас помочь ему решить эту задачу.

Входные данные

В первой строке находится одно целое число n ($3 \le n \le 2 \cdot 10^5$) — число полянок в парке.

В каждой из следующих (n - 1) строк находятся два целых числа x и y ($1 \le x, y \le n$) — номера двух полянок, соединенных очередной дорожкой. Гарантируется, что от любой полянки можно добраться до любой другой по дорожкам.

Выходные данные

В первой строке выведите одно целое число k — минимальное количество цветов, которое необходимо использовать Андрюше.

Во второй строке выведите n целых чисел, i-е из которых равняется цвету шарика, который нужно повесить на i-й полянке. Каждое из чисел должно быть в пределах от 1 до k.

Примеры

Входные данные

3

2 3

1 3

Выходные данные

3

1 3 2

Входные данные

5

2 3

5 3

4 3

1 3

Выходные данные

5

1 3 2 5 4

Входные данные

5

2 1

3 2

4 3

5 4

Выходные данные

3

1 2 3 1 2

Примечание

В первом примере из условия парк состоит из трех полянок, которые последовательно соединены: 1 → 3 → 2. Значит, цвета шариков на каждой полянке должны быть попарно различны.

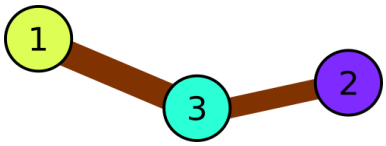


Иллюстрация к первому примеру.

Во втором примере в парке можно найти следующие тройки полянок, соединенных последовательно:

- 1 → 3 → 2
- 1 → 3 → 4

- 1 → 3 → 5
- 2 → 3 → 4
- 2 → 3 → 5
- 4 → 3 → 5

Отсюда мы видим, что каждая пара полянок лежит в какой-нибудь тройке, а значит цвета шариков на всех полянках должны быть попарно различны.

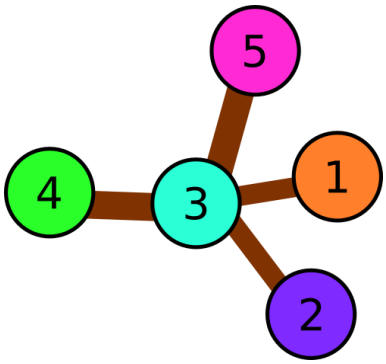


Иллюстрация ко второму примеру.

В третьем примере есть следующие тройки:

- 1 → 2 → 3
- 2 → 3 → 4
- 3 → 4 → 5

Это значит, что одного или двух цветов недостаточно, а для трех цветов ответ существует и приведен в примере.

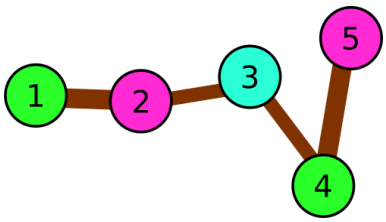


Иллюстрация к третьему примеру.

Разбор задачи:

Если вершина v имеет степень d, то ответ как минимум $d + 1$. Действительно, любые два соседа v лежат на общем пути длины 3, проходящем через v. Кроме этого, v лежит на общем пути из трех вершин с любым своим соседом (возможно, с использованием не-соседей). Значит, v и все его соседи должны иметь попарно разные цвета.

Покажем, что эта оценка является наилучшей. Построим раскраску дерева в $D + 1$ цвет, где D — максимальная степень вершины. Подвесим дерево за любую вершину, после чего раскрасим корень и всех его сыновей в цвета 1, 2, и так далее. Остальные вершины покрасим согласно следующему правилу: если вершина v имеет цвет x, а ее родитель имеет цвет y, то дети v получают последовательные цвета, начиная с 1 и пропуская цвета x и y. Несложно видеть, что цвета с номерами больше $D + 1$ никогда не понадобятся.

С точки зрения реализации данная процедура является обычным обходом в глубину.
Сложность: $O(n)$ времени и памяти.

D. Иннокентий и футбольная лига (максимум 1500 баллов)

ограничение по времени на тест 2 секунды
ограничение по памяти на тест 256 мегабайт
ввод стандартный ввод
вывод стандартный вывод

Иннокентий — президент только что созданной футбольной лиги в Байтландии. Первая задача, которая стоит перед ним — обеспечить трансляцию всех матчей лиги по телевидению. Как вы, возможно, знаете, во время футбольного матча на экране отображаются сокращенные названия команд и текущий счет. Конечно, будет странно, если сокращенные названия клубов соперников совпадут, поэтому Иннокентий должен придумать. как сократить название каждого из клубов лиги до трех букв так, чтобы у всех команд лиги были разные названия.

Название каждого клуба состоит из двух слов: названия команды и города, в котором располагается клуб, например, «DINAMO BYTECITY». Иннокентий не хочет сильно искажать название, поэтому он хочет выбрать сокращенное название для каждого из клубов таким образом, чтобы:

1. либо сокращенное название клуба совпадало с первыми тремя буквами названия команды, например, для вышеприведенный команды это «DIN»,
2. либо первые две буквы сокращенного названия клуба совпадали с первыми двумя буквами названия команды, а третья — с первой буквой города команды. Например, для вышеприведенный команды это «DIB».

Кроме этого, если для какой-то команды *x* выбран второй вариант названия, то не должно быть команды, у которой выбран первый вариант сокращенного названия, совпадающий с первым вариантом названия команды *x*. Например, если сокращенное название вышеприведенной команды это «DIB», то никакая команда, у которой выбран первый вариант сокращенного названия, не должна иметь сокращенного

названия «DIN». При этом возможно, что какая-то команда получит название «DIN», где «DI» — первые две буквы названия команды, а «N» — первая буква города. Конечно, никакие две команды не должны иметь одинакового сокращенного названия.

Помогите Иннокентию выбрать сокращенное название для каждой из команд. Если это невозможно, сообщите об этом. Если возможных ответов несколько, Иннокентия устроит любой из них. Если для некоторой команды совпадают оба варианта сокращенного названия, то Иннокентий все равно будет формально считать, что выбран только один из этих вариантов.

Входные данные

В первой строке находится единственное целое число n ($1 \le n \le 1000$) — число футбольных клубов в лиге.

В каждой из следующих n строк находятся два слова — название команды и название города очередного клуба. И название команды, и название города состоят из заглавных букв латинского алфавита и имеют длину не менее 3 и не более 20.

Выходные данные

Если выбрать сокращенные названия, удовлетворив требованиям Иннокентия, невозможно, выведите единственную строку «NO».

Иначе в первую строку выведите «YES». Далее выведите n строк, в каждой строке выведите выведите сокращенное название очередного клуба.

Выводите клубы в том же порядке, в каком они заданы во входных данных.

Если возможных ответов несколько, выведите любой.

Примеры

Входные данные

2

DINAMO BYTECITY

FOOTBALL MOSCOW

Выходные данные

YES

DIN

FOO

Входные данные

2

DINAMO BYTECITY

DINAMO BITECITY

Выходные данные

NO

Входные данные

3

PLAYFOOTBALL MOSCOW

PLAYVOLLEYBALL SPB

GOGO TECHNOCUP

Выходные данные

YES

PLM

PLS

GOG

Входные данные

3

ABC DEF

ABC EFG

ABD OOO

Выходные данные

YES

ABD

ABE

ABO

Примечание

В первом примере можно выбрать первый вариант названия для обеих команд.

Во втором примере невозможно подобрать названия, т. к. по условию нельзя у одной команды выбрать первый вариант названия, а у второй выбрать второй вариант названия, если первые варианты названий у этих команд совпадают.

В третьем примере можно выбрать вторые варианты названий у первых двух команд, и первый вариант — у третьей команды.

В четвертом примере обратите внимание на то, что разрешается, чтобы выбранное название некоторой команды *x* совпадало с первым вариантом названия *y*, если первый вариант названия *x* не совпадает с первым вариантом названия *y*.

Разбор задачи:

Будем обозначать a_i и b_i первый и второй вариант для названия *i*-ого клуба соответственно.

Если все a_i различны, мы можем использовать их все как названия. Иначе, предположим, что $a_i = a_j$ для каких-то клубов *i* и *j*, тогда их нельзя использовать одновременно. Кроме того, выбирать, например, a_i и b_j в такой ситуации также запрещено условием. Получается, что мы должны выбрать в качестве имен b_i и b_j.

Если теперь для какого-то еще клуба *k* мы имеем $a_k = b_i$, мы должно выбрать в качестве его названия b_k. Такие цепные зависимости можно обработать любым алгоритмом обхода. Если мы в какой-то момент будем обязаны использовать одинаковые имена, то решения нет, иначе после разрешения всех конфликтов получим правильное решение.

Сложность: $O(n)$ времени и памяти при аккуратной реализации (в данных ограничениях это было необязательно).

E. Подземная лабаратория (максимум 1750 баллов)

ограничение по времени на тест 1 секунда

ограничение по памяти на тест 256 мегабайт

ввод стандартный ввод

вывод стандартный вывод

В огромной подземной лаборатории злая корпорация Зонтик создает клонов для своих страшных экспериментов. Однажды компания клонировала мальчика по имени Андрюша, который был умнее своих сверстников. Андрюша понял, что вокруг творится что-то неладное. Он поднял клонов на борьбу против корпорации, и те начали искать выход из лаборатории. Компании ничего не оставалось, кроме как запустить процесс самоуничтожения подземного комплекса.

Лаборатория представляет собой связный граф из n вершин и m ребер. В некоторых вершинах этого графа k клонов Андрюши начинают поиск выхода из лаборатории. В процессе поиска каждую секунду они переходят по ребру в соседнюю вершину, причем в каждой вершине может одновременно находиться сколько угодно клонов. Каждый клон может в некоторый момент времени остановиться и прекратить поиски, но он обязательно должен их начать, то есть посетить хотя бы одну вершину. Более того, выход может находиться в произвольном месте лаборатории, поэтому клоны должны обойти граф целиком, то есть каждая вершина графа должна быть посещена хотя бы одним клоном хотя бы один раз.
Время клонов ограничено, и каждый из них сможет обойти не более $\lceil \frac{2n}{k} \rceil$ комнат до того, как лаборатория взорвется.

Ваша задача заключается в том, чтобы расставить клонов по вершинам графа и для каждого клона вывести путь, по которому он должен обходить

граф. При этом количество вершин в каждом пути должно быть не больше $\lceil \frac{2n}{k} \rceil$.

Входные данные

В первой строке находятся три целых числа n, m и k ($1 \le n \le 2 \cdot 10^5$, $n - 1 \le m \le 2 \cdot 10^5$, $1 \le k \le n$) — число вершин в графе, число ребер в графе и количество клонов.

В каждой из следующих m строк находятся два целых числа x_i и y_i ($1 \le x_i, y_i \le n$) — номера двух вершин, соединенных очередным ребром. В графе могут быть петли и кратные ребра.

Гарантируется, что граф связный.

Выходные данные

Выведите k строк. В начале i-й строки выведите целое число c_i ($1 \le c_i \le \lceil \frac{2n}{k} \rceil$) — количество вершин, которые посетит i-й клон, а затем выведите c_i целых чисел — номера вершин, которые он посетит, в порядке их посещения. Выводите вершину каждый раз, когда клон ее посещает, даже если он посещал ее до этого.

Гарантируется, что ответ существует.

Примеры

Входные данные

3 2 1

2 1

3 1

Выходные данные

3 2 1 3

Входные данные

5 4 2

1 2

1 3

1 4

1 5

Выходные данные

3 2 1 3

3 4 1 5

Примечание

В первом тесте есть всего 1 клон, он посещает вершины в порядке (2, 1, 3), что укладывается в ограничение 6 посещенных вершин.

Во втором тесте есть 2 клона, они посещают вершины в порядке (2, 1, 3) и (4, 1, 5), что укладывается в ограничение 5 посещенных вершин.

Разбор задачи:

Запустим обход в глубину из любой вершины графа и выпишем эйлеров обход — порядок, в котором обход посещает вершины, причем вершина v выписывается каждый раз, когда обход в нее заходит, в частности, после окончания каждого рекурсивного вызова, сделанного из v. Заметим, что эйлеров обход содержит $2n - 1$ запись и является корректным ответом для $k = 1$. Для общего случая k, разрежем эйлеров обход на k частей размера не более $\lceil 2n / k \rceil$, и выведем эти части в ответ. Отметим, что по условию

в каждой части должна быть хотя бы одна вершина, поэтому в пустые части построенного ответа необходимо дописать какую угодно вершину.
Сложность: $O(n + m)$ времени и памяти.

F. Аксель и Марстон в Битландии (максимум 2500 баллов)

ограничение по времени на тест 5 секунд
ограничение по памяти на тест 256 мегабайт
ввод стандартный ввод
вывод стандартный вывод

Два друга Аксель и Марстон вместе путешествуют по Битландии. В Битландии n городов, некоторые пары из которых соединены однонаправленными дорогами. Дороги в Битландии бывают пешеходные и велосипедные. В Битландии может быть несколько дорог между каждой парой городов, и бывают даже дороги из города в него же. Однако, никакие две дороги в Битландии не могут иметь одинаковые начало, конец и тип одновременно.

Друзья находятся в городе 1 и хотят составить маршрут путешествия. Аксель любит ходить пешком, а Марстон предпочитает велосипед. Чтобы маршрут получился разнообразным и интересным для каждого из друзей, они выбирают порядок чередования типов пройденных дорог следующим образом:

- Маршрут начинается с пешеходной дороги.
- Пусть известное начало маршрута записано в виде строки s из букв П (пешеходная дорога) и В (велосипедная дорога). Тогда к маршруту s

дописывается строка $\overline{s}$, где $\overline{s}$ означает строку *s*, в которой тип каждой дороги заменен на противоположный (все пешеходные дороги на велосипедные, и наоборот).

Первые несколько шагов построения маршрута будут выглядеть так: П, ПВ, ПВВП, ПВВПВППВ, ПВВПВППВВПППВПВВП, и так далее.

Далее, друзья начинают движение в городе 1 по дорогам Битландии, каждый раз выбирая тип очередной дороги в соответствии с очередным символом маршрута. Если же подходящую дорогу выбрать не удается, друзья заканчивают свое путешествие и летят домой.

Помогите друзьям определить максимально возможную продолжительность своего путешествия, если они будут следовать построенной последовательности типов дорог. Если друзья могут найти путь, состоящий более, чем из 10^{18} дорог, вместо длины выведите -1.

Входные данные

Первая строка содержит два целых числа n и m ($1 \le n \le 500$, $0 \le m \le 2n^2$) — количество городов и дорог в Битландии соответственно.

Следующие m строк описывают дороги. В i-ой из этих строк записано три целых числа v_i, u_i и t_i ($1 \le v_i, u_i \le n$, $0 \le t_i \le 1$), где v_i и u_i — номера городов, являющихся началом и концом i-ой дороги, а t_i задает тип i-ой дороги (0 — пешеходная дорога, 1 — велосипедная дорога). Гарантируется, что для каждой пары различных чисел i, j, таких что $1 \le i, j \le m$, либо $v_i \ne v_j$, либо $u_i \ne u_j$, либо $t_i \ne t_j$.

Выходные данные

Если друзья могут найти подходящий путь длины строго большей, чем 10^{18}, выведите одно число -1. В противном случае выведите максимальную длину подходящего пути, то есть наибольшее количество дорог, которое могут пройти друзья.

Примеры

Входные данные

2 2

1 2 0

2 2 1

Выходные данные

3

Входные данные

2 3

1 2 0

2 2 1

2 2 0

Выходные данные

-1

Примечание

В первом примере путь длины 3 можно получить, пройдя один раз по дороге 1 из города 1 в город 2, после чего дважды по дороге 2 из города 2 в него же.

Во втором примере мы можем получить путь сколь угодно большой длины, сперва пройдя по дороге 1, а после этого выбирая дорогу 2 или 3 в зависимости от очередного типа дороги.

Разбор задачи:

Будем обозначать A_i двоичную строку, полученную i повторений обращения и приписывания, например, $A_0 = 0$, $A_1 = 01$, и т.д. Также обозначим $B_i = \overline{A_i}$. По определению $A_{i+1} = A_i B_i$, и $B_{i+1} = B_i A_i$.

Насчитаем матрицы P_k и Q_k, элементы P_k / $Q_k(v, u)$ в которых равны 1 для пар вершин v, u, таких что из v в u есть путь, соответствующий строке A_k/B_k. Матрицы P_0 и Q_0 — это, очевидно, просто матрицы смежности с 0- и 1-ребрами соответственно.

Дальше заметим, что $P_{k+1}(v, u) = 1$ тогда и только тогда, когда для какой-то вершины w выполнено $P_k(v, w) = Q_k(w, u) = 1$, и подобный критерий работает для $Q_{k+1}(v, u)$. Таким образом, мы можем посчитать P_{k+1} и Q_{k+1}, используя P_k и Q_k, за время $O(n^3)$ (по сути, пересчет заключается в перемножении битовых матриц: $P_{k+1} = P_k Q_k$, $Q_{k+1} = Q_k P_k$).

Теперь при помощи P_k и Q_k найдем ответ. Будем хранить текущий максимальный ответ L, и множество вершин S, достижимых из вершины 1 по какому-то корректному пути длины L. Будем перебирать k в порядке убывания с некоторого значения k_0, и проверять, можно ли увеличить L на 2^k. После L-ой позиции следующие 2^k символов образуют строку A_k либо B_k в зависимости от четности количества единиц в битовой записи L. Пусть S' — множество вершин, достижимых из S по пути, соответствующему A_k / B_k. Если S' непустое, увеличим L на 2^k, и присвоим $S = S'$, иначе, ничего не будем делать. В конце концов, L будет максимальной длиной пути, если она меньше, чем 2^{k_0}.

Возьмем $k_0 = 60$, поскольку нас не интересует точное значение ответа, если он больше 2^{60}. Получилось решение за время $O(k_0 n^3)$, что слишком медленно. Воспользуемся битовой оптимизацией умножения матриц, это ускорит решение в ~ 64 раза.

Сложность: $O(k_0 n^3 / w)$ времени и $O(k_0 n^2)$ памяти, где $k_0 = \log_2 10^{18}$, $w = 64$ — длина машинного слова в битах.

G. Андрюша и живые барьеры (максимум 3000 баллов)

ограничение по времени на тест 4 секунды

ограничение по памяти на тест 256 мегабайт

ввод стандартный ввод

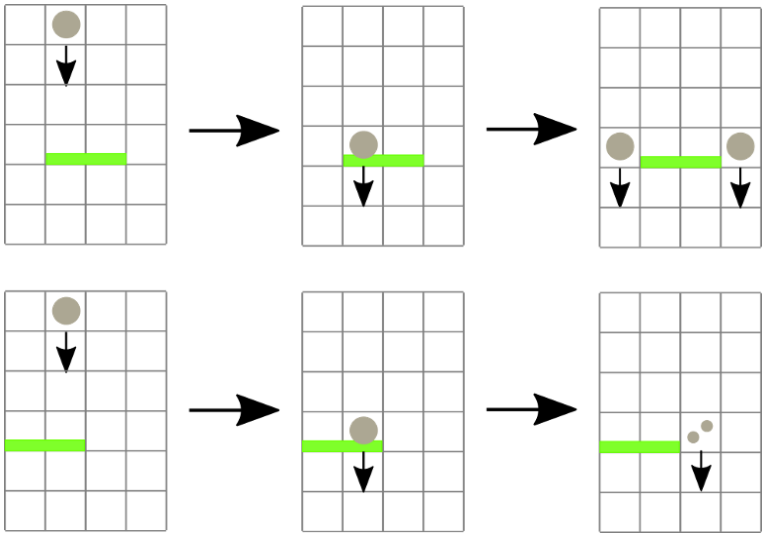
вывод стандартный вывод

Недавно Андрюша нашел удивительный игровой автомат. Он представлял из себя доску, расположенную вертикально, разбитую на квадраты. Всего на доске было w столбцов, пронумерованных от 1 до w слева направо, и h строк, пронумерованных снизу вверх от 1 до h.

Кроме того, в некоторых строках автомата были расположены перегородки. Всего перегородок было n, и i-я из них располагалась в строке ui, занимая все клетки в столбцах с li по ri. Никакие две перегородки не находились в одной строке, Андрюша это точно запомнил.

Кроме того, в каждой строке была хотя бы одна клетка, в которой не было перегородки.

В любой столбец автомата можно было сверху бросить шарик. В таком случае шарик начинал падать вниз, пока не натыкался на какую-либо перегородку или не выпадал из автомата снизу. Если шарик натыкался на перегородку, то он исчезал, а вместо него возникали два шарика слева и справа от перегородки, и продолжали падать по тем же правилам. Если слева или справа от перегородки находился край доски, то оба шарика появлялись с той стороны, где нет края. При этом могло быть, что в какой-то клетке оказывались два или более шарика одновременно, это никак не влияло на их движение. В конце все шарики выпадали из автомата снизу.



Примеры того, как шарики взаимодействуют с барьерами.

Особенность автомата была в том, что иногда шарики проходили сквозь перегородки, как через свободные клетки. Это связано с тем, что перегородки были живыми и боялись шариков, падающих с большой высоты. А именно, если на перегородку номер *i* падал шарик, который

перед этим падал более, чем *si* клеток (то есть шарик появился в строке с номером строго больше, чем *ui* + *si*), то перегородка пропускает этот шарик, как будто ее нет. Если шарик был брошен в автомат сверху, считайте, что он появился на высоте (*h* + 1).

Андрюша помнит, что он бросил по одному шарику в каждый из столбцов. Найдите для него общее количество шариков, выпавших снизу. Так как это число может быть большим, выведите остаток от деления этого числа на 109 + 7.

Входные данные

В первой строке находятся три целых числа *h*, *w* и *n* ($1 \le h \le 109$, $2 \le w \le 105$, $0 \le n \le 105$) — число строк, число столбцов и число перегородок в автомате.

В каждая из следующих *n* строк находятся четыре целых числа. В *i*-й из этих строк находятся числа *ui*, *li*, *ri*, *si* ($1 \le ui \le h$, $1 \le li \le ri \le w$, $1 \le si \le 109$) — строка, в которой находится *i*-я перегородка. столбцы, в которых находятся края этой перегородки, и максимальная высота, с которой перегородка не пропускает шарики. Гарантируется, что в каждой строке есть хотя бы одна клетка, в которой нет перегородки, а также то, что все *ui* различны.

Выходные данные

Выведите единственное число — остаток от деления суммарного числа выпавших шариков на 109 + 7.

Примеры

входные данные

10 5 1

3 2 3 10

выходные данные

7

входные данные

10 5 2

3 1 3 10

5 3 5 10

выходные данные

16

входные данные

10 5 2

3 1 3 7

5 3 5 10

выходные данные

14

входные данные

10 15 4

7 3 9 5

6 4 10 1

1 1 4 10

4 11 11 20

выходные данные

53

Примечание

В первом примере один барьер: если бросить шарик во второй или третий столбец, то выпадет два шарика, иначе — один. Итого ответ 7.

Во втором примере количество выпавших шариков равно 2, 2, 4, 4, 4 при бросании шариков в столбцы слева направо соответственно.

В третьем примере количество выпавших шариков равно 1, 1, 4, 4, 4 при бросании шариков в столбцы слева направо соответственно. Обратите внимание, первый барьер пропускает шарики, падающие на него напрямую, но не пропускает те, что падают на него со второго барьера.

В четвертом примере количество выпавших шариков равно 2, 2, 6, 6, 6, 6, 6, 6, 6, 1, 2, 1, 1, 1, 1 при бросании шариков в столбцы слева направо соответственно. Случай, когда шарик брошен в седьмой столбец, рассмотрен на рисунке ниже.

Разбор задачи:

Решение 1: Будем двигать сканирующую прямую снизу вверх. Назовем *i*-ую перегородку активной, если текущая координата *y*удовлетворяет $ui \le y \le ui + si$. Если мы хотим узнать результат бросания шарика в столбец *x* в текущий момент, нам надо определить самую высокую из активных в данный момент перегородок.

Заведем дерево отрезков с множеством активных отрезков, покрывающих каждую вершину. Чтобы добавить новую активную перегородку, разобьем

ее на $O(\log w)$ вершин дерева и в каждую из них добавим запись об этой перегородке; чтобы удалить эту перегородку, удалим все эти записи. Если мы хотим узнать самую высокую из активных сейчас перегородок, пройдемся по вершинам дерева, покрывающим *x*, и в каждой из них посмотрим на самую высокую перегородку в множестве. Также, для каждой перегородки будем хранить результат (количество получающихся в итоге шариков) от попадания в нее одного шарика; чтобы найти этот результат, нужно в момент активации перегородки сделать два запроса в дерево (для шариков, падающих слева и справа). Получаем решение со сложностью $O((n + w) \log^2 w)$ (со вторым логарифмом от времени работы std::set).

Решение 2: На этот раз пойдем сверху вниз, и будем хранить позиции всех шариков, которые мы бросаем. Если какая-то группа шариков оказывается в одной точке, мы объединяем их вместе и запоминаем размер группы. В каждой колонке будем хранить все группы шариков в стеке с более низкими группами на вершине.

Пусть мы встретили перегородку [*l*, *r*]. Для каждой колонки из диапазона [*l*, *r*] в эту перегородку попадут несколько нижних групп, и все они превратятся в (не более чем) две группы по краям перегородки. Заведем дерево отрезков размера *w*, и для каждой колонки будем хранить в нем высоту самой низкой группы в этой колонке. Теперь, пока минимум на отрезке [*l*, *r*], выкинем соответствующую группу из стека (не забыв обновить дерево). Наконец, создадим новые группы и положим в нужные стеки. Оставшиеся после обработки всех перегородок группы шариков выпадут снизу.

Стандартное амортизационное рассуждение показывает, что всего будет проделано $O(n + w)$ операций, тем самым сложность решения составляет $O((n+w)\log w)$.

H. Автобусы и интранет (максимум 3500 баллов)

ограничение по времени на тест 10 секунд

ограничение по памяти на тест 256 мегабайт

ввод стандартный ввод

вывод стандартный вывод

В городе $\mathbb{N}$ запустили автобусный маршрут, представляющий собой замкнутую ломаную на плоскости, все звенья которой параллельны осям координат. На маршруте будет работать m автобусов, двигающихся по циклу вдоль ломаной в одном направлении с одинаковой постоянной скоростью (временем остановок автобусов в рамках данной задачи можно пренебречь).

Автобусы начинают движение по маршруту в первой вершине ломаной с равными промежутками. Пусть T — это время, за которое один автобус проезжает по кругу весь маршрут. Тогда автобус 1 начинает движение в момент времени 0, автобус 2 — в момент времени T / m, третий — в момент времени $2T / m$, и т.д.; наконец, автобус m стартует в момент $(m - 1)T / m$. Таким образом, интервалы между всеми парами соседних автобусов (в том числе, между последним и первым) одинаковы.

Автобусы могут обмениваться информацией при помощи беспроводных передатчиков одинаковой мощности. Если мощность передатчиков на автобусах равна *D*, то обмен информацией возможен между автобусами на расстоянии *D* или меньше.

Кроме этого, автобусы оснащены распределенной системой слежения за маршрутом. Для того, чтобы все автобусы двигались строго по графику, система должна периодически синхронизировать данные на всех автобусах. В момент синхронизации автобус 1 обменивается информацией с автобусом 2, автобус 2 с автобусом 3, и т.д., кроме этого, автобус *m* обменивается информацией с автобусом 1.

Вам, как сотруднику аналитического отдела, поставили задачу найти минимальное значение *D*, при котором синхронизацию возможно проводить хотя бы время от времени.

Входные данные

В первой строке записано два целых числа *n* и *m* ($2 \le n, m \le 105$) — количество вершин ломаной и количество автобусов соответственно.

Следующие *n* строк описывают вершины ломаной в порядке ее обхода. Каждая из этих строк содержит два целых числа *xi*, *yi*($-1000 \le xi, yi \le 1000$) — координаты очередной вершины.

Гарантируется, что каждое звено ломаной (в том числе, между последней и первой вершиной) параллельно одной из осей координат. Кроме этого, никакие две последовательные вершины ломаной не совпадают. Ломаная может иметь самопересечения и проходить по одному отрезку несколько раз.

Выходные данные

Выведите одно вещественное число — ответ на задачу. Ответ будет принят, если абсолютная или относительная погрешность не превосходит 10 - 6.

Примеры

входные данные

```
4 2
0 0
0 1
1 1
1 0
```

выходные данные

```
1.000000000
```

входные данные

```
2 2
0 0
1 0
```

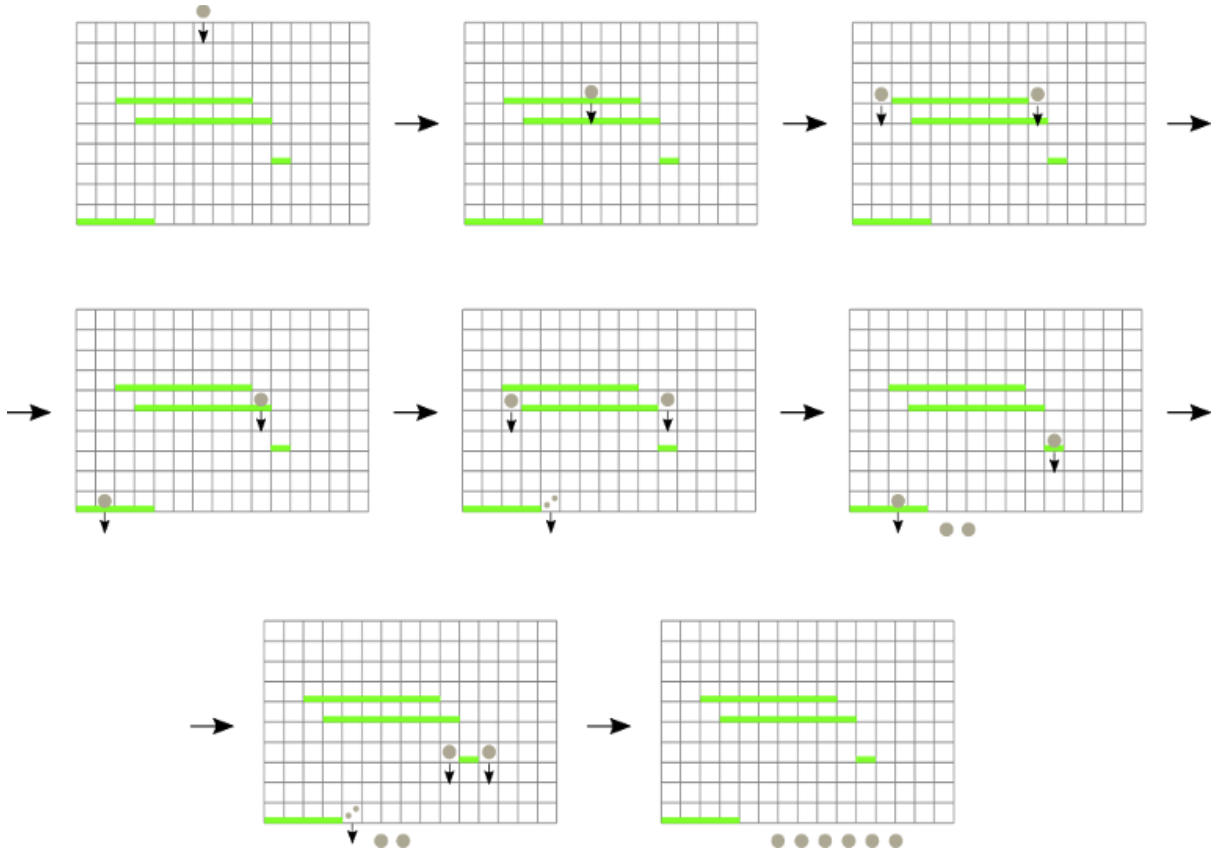
выходные данные

```
0.000000000
```

Примечание

Пусть все автобусы проезжают 1 единицу расстояния в секунду. В первом примере, через 0.5 секунды автобусы окажутся на расстоянии 1, поэтому можно выбрать $D = 1$.

Во втором примере, через 0.5 секунды оба автобуса окажутся в точке (0.5, 0), поэтому можно выбрать $D = 0$.



Результат, если бросить шарик в седьмой столбец.

Разбор задачи:

Будем делать бинарный поиск по ответу. Внутри нужно проверить, есть ли такой момент времени, что все пары автобусов 1 и 2, 2 и 3, ..., n и 1 одновременно находятся на расстоянии не больше x.

Пусть $p(t)$ — положение автобуса 1 (который отправился в момент 0) через время t. Назовем момент времени t *хорошим*, если выполнено $||p(t + T / m) - p(t)|| \le x$, где $||a - b||$ означает расстояние между точками a

и b. Если есть такой момент t, что t, $t + T / m$, ..., $t + (m - 1)T / m$ все являются хорошими, то ответ не меньше x.

Построим двумерный график вектора $p(t + T / m) - p(t)$ по времени, для этого будем следить за тем, на каких сторонах ломаной находятся точки $p(t)$ и $p(t + T / m)$. Если каждая точка двигается по своей стороне, то вектор $p(t + T / m) - p(t)$ со временем меняется линейно, тем самым график также является кусочно линейным. Моментов, когда какая-то из точек переходит со стороны на сторону, всего $O(n)$, поэтому график можно построить за время $O(n)$ двумя указателями.

Теперь найдем множество хороших моментов t для фиксированного x. Сделаем это отдельно для каждого отрезка в графике разностей, потом составим ответы вместе. С точки хрения разности $q = p(t + T / m) - p(t)$ у нас есть условие $||q|| \le x$, где qпробегает определенный отрезок. Это стандартная задача нахождения пересечения отрезка с кругом, которую можно решать многими способами, например, решить квадратное уравнение или повернуть определенный вектор на нужный угол. В любом случае, в результате получаем некоторый отрезок времени либо пустое множество. Отметим, что на некоторые участках графика q может оставаться постоянным, этот случай лучше обработать отдельно.

Чтобы найти момент, когда $p(t)$, $p(t + T / m)$, и т.д. являются хорошими, разрежем отрезок $[0, T]$ на m равных частей и «наложим» их друг на друга, тем самым хорошие отрезки тоже придется разрезать на части. Теперь подходящий момент t — это точка, покрытая m раз. Поиск такой точки — стандартное применение сканирующей прямой.

Сложность: $O(n \log \varepsilon^{-1})$, где ε — требуемая относительная точность.