Технокубок 2017/2018. Условия и разбор задач.

A. Чемпионат мира

ограничение по времени на тест

1 секунда

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Заключительная часть чемпионата мира по футболу проводится по олимпийской системе, также называемой плей-офф.

Всего в этой части турнира принимает участие n команд, пронумерованных от 1 до n. Проводится несколько раундов, в каждом раунде оставшиеся команды располагаются в порядке увеличения номеров, затем первая играет со второй, третья — с четвёртой, пятая — с шестой и так далее. Гарантируется, что в каждом раунде участвует четное число команд. Команда-победитель в каждой игре проходит в следующий раунд, проигравшая команда выбывает из турнира, ничьих не бывает. В последнем раунде принимают участие две команды, этот раунд называется финалом, а команда-победитель объявляется чемпионом мира, на этом турнир заканчивается.

Аркадий хочет, чтобы в финале сыграли две определенные команды. К сожалению, номера команд уже определены, и может получиться так, что эти команды не могут выйти в финал одновременно, так как в лучшем случае встретятся на каком-то раунде до финала. Определите, в каком раунде возможна встреча команд с номерами a и b.

Входные данные

Единственная строка содержит три целых числа n, a и b ($2 \le n \le 256$, $1 \le a, b \le n$) — общее число команд и номера команд, которыми интересуется Аркадий.

Гарантируется, что n таково, что в каждый раунд проходит четное число команд, а a и b — различные числа.

Выходные данные

Выведите в единственной строке «`Final!`» (без кавычек), если команды a и b могут встретиться в финале.

Иначе выведите одно целое число — номер раунда, в котором могут встретиться команды a и b. Раунды нумеруются с 1.

Примеры

входные данные

```
4 1 2
```

выходные данные

```
1
```

входные данные

```
8 2 6
```

выходные данные

```
Final!
```

входные данные

```
8 7 5
```

выходные данные

```
2
```

Примечание

В первом примере команды с номерами 1 и 2 встретятся между собой в первом же раунде.

Во втором примере команды 2 и 6 могут встретиться между собой только в третьем раунде (который будет финальным), в том случае, если они победят своих оппонентов в первом и втором раундах.

В третьем примере команды с номерами 7 и 5 могут встретиться во втором раунде, если обыграют своих оппонентов в первом раунде

.

944A - Чемпионат мира

Изначально нужно понять следующий факт. Так как количество команд в каждом раунде было чётным, то n обязательно является степенью двойки.

Будем решать задачу в 0-индексации команд, то есть сразу уменьшим a и b на единицу.

Будем узнавать номер матча, в котором играют команды с изначальным номерами a и b в каждом раунде. Команда a будет играть в матче номер $a / 2$, а команда b в матче номер $b / 2$.

Если $a / 2 = b / 2$, то эти команды будут играть в одном матче, и нужно вывести номер текущего раунда в качестве ответа. Если при этом количество оставшихся команд равно двум, то это будет финальный матч.

Если же номера матчей не совпадают, то нужно перейти к следующему раунду. При этом номер команды a станет равным $a / 2$, а номер команды b станет равным $b / 2$. Количество команд которые перейдут в следующий раунд уменьшится вдвое, то есть нужно $n = n / 2$. Этот процесс всегда конечен, так как рано или поздно команд останется всего 2 и в этом раунде состоится всего один матч, который будет финалом турнира.

B. Лабораторная работа

ограничение по времени на тест

1 секунда

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Аня и Кирилл делают лабораторную работу по физике. В одном из заданий необходимо n раз измерить некоторую величину, чтобы уменьшить ошибку, и вычислить среднее значение.

Кирилл выполнил необходимые измерения, получив целочисленные значения $x_1, x_2, \ldots, x_n$. Важно отметить, что разброс измеренных значений не очень большой, а именно, максимальное измеренное значение **не более чем на** 2 превосходит минимально измеренное значение.

Ане лень делать измерения, но просто так списать все измерения у Кирилла она не может, ведь величина ошибки — случайная величина, и если все измерения совпадут, преподаватель заподозрит неладное. Аня хочет написать в своей работе такие целочисленные величины измерений $y_1, y_2, \ldots, y_n$, что выполняются следующие условия:

- среднее значение среди величин $x_1, x_2, \ldots, x_n$ точно равно среднему значению $y_1, y_2, \ldots, y_n$;
- все измерения в работе Ани лежат в тех же пределах, что и измерения Кирилла, то есть максимальное значение среди измерений Ани не превосходит максимальное значение среди измерений Кирилла, а минимальное значение среди измерений в работе Ани не меньше, чем минимальное значение среди измерений Кирилла;
- количество совпадающих измерений у Ани и Кирилла — минимально возможное при выполнении предыдущих условий. Формально, учитель смотрит на все измерения Ани по очереди, если такое же измерение есть у Кирилла и оно еще не зачеркнуто, он вычеркивает одно это измерение у Ани и одно из таких измерений у Кирилла. Количеством совпадающих измерений называется количество **зачеркнутых** измерений у Ани.

Помогите Ане написать такой набор измерений, чтобы выполнялись условия, перечисленные выше.

Входные данные

В первой строке следует целое число n ($1 \le n \le 100\,000$) — количество измерений, которые сделал Кирилл.

Во второй строке следует последовательность $x_1, x_2, ..., x_n$ ($-100\,000 \le x_i \le 100\,000$) — результаты измерений Кирилла. Гарантируется, что разность между максимальным и минимальным значением среди чисел $x_1, x_2, ..., x_n$ не превосходит 2.

Выходные данные

В первую строку выведите минимальное количество совпадающих измерений.

Во вторую строку выведите n целых чисел $y_1, y_2, ..., y_n$ — результаты измерений Ани. Числа могут быть выведены в любом порядке. Помните, что минимальное из измерений Ани должно быть не меньше, чем минимум из измерений Кирилла, а максимальное из измерений Ани должно быть не больше, чем максимум из измерений Кирилла.

Если существует несколько решений, выведите любое.

Примеры

входные данные

```
6
-1 1 1 0 0 -1
```

выходные данные

```
2
0 0 0 0 0 0
```

входные данные

```
3
100 100 101
```

выходные данные

```
3
101 100 100
```

входные данные

```
7
-10 -9 -10 -8 -10 -9 -9
```

выходные данные

```
5
-10 -10 -9 -9 -9 -9 -9
```

Примечание

В первом примере Аня может записать нули в качестве каждого из результатов измерений. Тогда средняя величина её измерений будет равна средней величине измерений Кирилла, а совпадающих измерений будет всего два.

Во втором примере Аня должна записать два результата измерений, равные 100, и одно 101 (сами измерения она может записать в любом порядке), потому что только в этом случае средняя величина её измерений будет равна средней величине измерений Кирилла. Таким образом, все три измерения будут совпадать.

В третьем примере количество совпадающих измерений равно 5.

944B - Лабораторная работа

Так как нужно, чтобы среднее значение измерений Ани было равно среднему значению измерений Кирилла, то и сумма всех измерений Ани должна быть равна сумме всех измерений Кирилла.

Пусть минимальное число в измерениях Кирилла равно min, а максимальное max. Тогда, если $(max - min)$ меньше либо равно единице, то Аня не сможет записать ни одного измерения, которого не будет у Кирилла, поэтому все её измерения совпадут с его измерениями.

Остаётся случай когда $(max - min) = 2$. По условию, каждое измерений Ани должно быть не менее min и не более max. Тогда переберём сколько из её измерений будет равно min от 0 до n. Тогда количество измерений, равных $(min + 1)$ и max определяется однозначно.

Пусть sum — это необходимая сумма всех измерений, которая равна сумме всех измерений Кирилла, а $cntMin$ — это текущее количество минимумов, которые Аня может записать в качестве результатов измерений. Тогда Ане нужно набрать за оставшиеся измерения сумму $leftSum = sum - cntMin \cdot min$.

Минимальная сумма, которую может набрать Аня за оставшиеся измерения $minSum = (n - cntMin) \cdot (min + 1)$, а максимальная — $maxSum = (n - cntMin) \cdot max$. Тогда, если $leftSum < minSum$ или $leftSum > maxSum$, то нельзя взять $cntMin$ минимальных значений и получить нужную сумму.

В противном случае, Аня должна записать измерения, равные $(min + 1)$ в количестве $(leftSum - minSum)$ штук, а все оставшиеся измерения будут равны значению max. После этого нужно обновить ответ количеством совпадающих значений min, $(min + 1)$ и max в измерениях Ани и в измерениях Кирилла.

После обновления ответа переходим в переборе к следующему значению $cntMin$.

C. Необычная яблоня

ограничение по времени на тест

1 секунда

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

У Аркадия в саду есть одна необычная яблоня, с которой иногда необходимо собирать яблоки. Так как яблоня необычная, на ней есть n соцветий, они пронумерованы от 1 до n. Соцветие номер 1 находится около корня дерева, любое другое соцветие с номером i ($i > 1$) расположено на верхнем конце ветки, нижний конец которой расположен в соцветии p_i, при этом $p_i < i$.

Когда яблоки созревают, одновременно появляется ровно по одному яблоку в каждом соцветии. В тот же момент все яблоки начинают скатываться вниз по веткам к корню дерева. Каждую секунду все яблоки, кроме находящегося в первом соцветии, одновременно скатываются на одну ветку ближе к корню, то есть, например, из соцветия a яблоко попадет в соцветие p_a. Яблоки, находившиеся в первом соцветии, забирает Аркадий. Яблоня необычная, поэтому, если в какой-то момент в одном соцветии оказываются два яблока, они аннигилируют. Так происходит с каждой парой, например, если в соцветие попадет одновременно 5яблок, то останется только одно яблоко, а если в соцветие попадет одновременно 8 яблок, то не останется ни одного яблока. Таким образом, в каждом соцветии в любой момент времени находится не более одного яблока.

Помогите Аркадию, подсчитайте, сколько яблок он сможет забрать из первого соцветия за один урожай.

Входные данные

В первой строке следует целое число n ($2 \le n \le 100\,000$) — количество соцветий.

Во второй строке следует последовательность целых чисел $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), состоящая из $n - 1$ числа, где p_i равно номеру соцветия, в которое скатывается яблоко из соцветия i.

Выходные данные

Выведите количество яблок, которые Аркадий сможет забрать из первого соцветия за один урожай.

Примеры

входные данные

```
3
1 1
```

выходные данные

```
1
```

входные данные

```
5
1 2 2 2
```

выходные данные

```
3
```

входные данные

```
18
1 1 1 4 4 3 2 2 2 10 8 9 9 9 10 10 4
```

выходные данные

```
4
```

Примечание

В первом примере Аркадий соберет всего одно яблоко, которое изначально находилось в соцветии номер 1. Через секунду в это соцветие попадёт еще два яблока из соцветий 2 и 3, но они аннигилируют между собой и Аркадий не сможет собрать ни одно из них.

Во втором примере Аркадий соберет три яблока. Первым он соберет то яблоко, которое изначально находится в соцветии один. Через секунду в соцветие 1 попадет яблоко из соцветия 2 (которое Аркадий тоже соберет), а в соцветия 2 попадут три яблока из соцветий 3, 4 и 5, два из которых аннигилируют между собой, и в соцветии 2 останется всего одно яблоко, которое еще через секунду попадёт в соцветие 1, и Аркадий его соберёт.

944C - Необычная яблоня

Рассмотрим яблоки, которые могут попасть в корневую вершину дерева в момент времени t. Так как задано корневое дерево, то это те яблоки, которые находились изначально на расстоянии t от корня.

Поймем следующий факт. Предположим, что яблоки аннигилируют только тогда, когда попадают в корень. Это действительно так, ведь количество яблок, оставшихся в корне дерева в момент времени t зависит только от чётности количества яблок, попавших в него. Поэтому если два яблока одновременно оказались в какой-то вершине, отличной от корня, то можно считать, что они аннигилируют в корневой вершине позднее, ведь это не изменит

чётности количества яблок, которые попадают в корень дерева в каждый из моментов времени.

Поэтому посчитаем для каждого момента времени t величину cnt_t — сколько яблок окажется в корне в момент времени t. Это можно сделать с помощью обычного обхода в ширину. Также можно использовать обход в глубину, так как заданный граф ориентированный и обладает таким свойством, что из каждой вершины выходит ровно одно ребро (за исключением корня).

Таким образом, ответ на задачу это сумма всех $cnt_t \bmod 2$ ($a \bmod b$ означает взятие числа a по модулю b) для всех t от 0 до d, где d — максимальное расстояние от какой-то из вершин дерева до корня.

D. Игра со строкой

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

Вася и Коля играют в игру со строкой по следующим правилам. Сначала Коля загадывает строку s, состоящую из строчных латинских букв, и равновероятно выбирает из отрезка $[0, len(s) - 1]$ целое число k. Он сообщает строку s Васе, а затем циклически сдвигает её на k символов влево, то есть получает новую строку $t = s_{k+1}s_{k+2}...\ s_n s_1 s_2 ...\ s_k$. Вася не знает ни числа k, ни итоговой строки t, но хочет угадать число k. Для этого он может попросить Колю открыть первую букву получившейся строки, а затем, увидев её, еще одну букву на некоторой позиции, которую Вася может выбрать.

Вася уже понял, что не может гарантировать себе победу, однако, он хочет узнать вероятность своего выигрыша при оптимальной игре. Вам необходимо помочь ему в этом.

Заметим, что целью Васи является однозначное определение числа k, значит, если после открытия второй буквы остается не менее двух циклических сдвигов исходной строки s, удовлетворяющих известной информации, Вася считается проигравшим. Конечно же, в каждый момент игры Вася пытается максимизировать вероятность своего выигрыша, насколько это возможно.

Входные данные

Единственна строка содержит строку s длины l $(3 \le l \le 5000)$, состоящую из строчных латинских букв.

Выходные данные

Выведите одно вещественное число — ответ на задачу. Ответ будет считаться верным, если его абсолютная или относительная ошибка не будет превосходить 10^{-6}.

Формально, пусть ваш ответ равен a, а ответ жюри — b. Ваш ответ считается правильным, если $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Примеры

входные данные

```
technocup
```

выходные данные

```
1.000000000000000
```

входные данные

```
tictictactac
```

выходные данные

```
0.333333333333333
```

входные данные

```
bbaabaabbb
```

выходные данные

```
0.100000000000000
```

Примечание

В первом примере после открытия первой буквы Вася может попросить открыть вторую букву, и после этого циклический сдвиг определяется однозначно.

Во втором примере если первой буквой в циклическом сдвиге t будет «t» или «c», то у Васи не получится угадать сдвиг, открыв лишь одну другую букву. В то же время, если первой буквой будет «i» или «a», то, открыв четвертую букву, можно однозначно определить циклический сдвиг.

944D - Игра со строкой

Переберем, какая буква c_1 будет первой в строке t. Переберем, про какую по номеру d вторую букву спросит Вася. Теперь, если пара букв (c_1, c_2) встречается единственный раз на расстоянии d, то, если во второй раз откроется буква c_2, то Вася сможет однозначно определить сдвиг.

Рассмотрим все буквы на расстоянии d от всех букв c_1, посчитаем для каждого символа c_2 количество таких букв. Это можно сделать за $O(cnt(c_1))$, где $cnt(c_1)$ — количество букв c_1 в исходной строке. Теперь, если мы выберем такое d при открытии буквы c_1, которое максимизирует число p уникальных пар (c_1, c_2) на расстоянии d, то это и будет оптимальным выбором d, а условная вероятность победы при первой букве c_1 равна $p / cnt(c_1)$.

Осталось лишь просуммировать условные вероятности для различных букв c_1. Вероятность выпадения буквы c_1 равна $cnt(c_1) / n$, поэтому итоговая вероятность равна $\sum \frac{p}{cnt(c_1)} \cdot \frac{cnt(c_1)}{n} = \frac{\sum p}{n}$.

E. Федя не врёт!

ограничение по времени на тест

1 секунда

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Мальчик Федя очень любит рисовать. Больше всего он любит рисовать отрезки с целочисленными координатами внутри своего любимого отрезка $[1;m]$. Однажды Федя нарисовал несколько отрезков внутри своего любимого отрезка и заметил одну интересную особенность: не существует точки, принадлежащей сразу всем отрезкам. Заметив это, Федя очень обрадовался и захотел похвастаться об этом своему другу Саше.

Саша знает, что Федя любит прихвастнуть, поэтому с осторожностью относится к его заявлениям, никогда не доверяя на слово. Федя хочет доказать, что иногда ему всё же можно верить, поэтому он решил убедить Сашу в том, что на его рисунке действительно не существует точки, принадлежащей всем отрезкам. Но Федя – весьма ленивый человек, поэтому он не хочет сообщать Саше координаты концов каждого отрезка. Более того, ему лень говорить Саше, сколько отрезков он нарисовал.Вместо этого он предложил Саше задать несколько вопросов вида: «Скольким отрезкам принадлежит точка с целочисленной координатой x_i?», пообещав дать на них честные ответы.

Ребята очень ценят своё время, поэтому они просят вас посчитать, какое максимальное количество вопросов Саша может задать Феде, что располагая только этими Федиными ответами, Саша всё ещё не может быть полностью уверен в том, что Федя его не обманул. Учтите, что Саша не знает, сколько отрезков нарисовал Федя. Разумеется, Саша смышлёный малый и не будет спрашивать дважды про одну и ту же точку.

Входные данные

В первой строке входных данных находятся два целых числа n и m ($1 \le n, m \le 100\,000$) — количество отрезков, которые есть на Федином рисунке и максимальная координата точки, которую можно выбрать в наборе.

В i-й из следующих n строк записаны два целых числа l_i и r_i ($1 \le l_i \le r_i \le m$) — левая и правая границы i-го отрезка на рисунке. Обратите внимание, что левая и правая границы отрезка могут совпадать.

Гарантируется, что не существует точки, принадлежащей сразу всем нарисованным отрезкам.

Выходные данные

В единственной строке выходного файла выведите число k — размер максимального множества пар $(x_i, cnt(x_i))$, где все x_i – различны, $1 \le x_i \le m$, а $cnt(x_i)$ — количество отрезков, которым принадлежит точка с координатой x_i, такого что, зная только это множество пар (и не зная число n), нельзя гарантированно утверждать, что в исходном рисунке не существует точки, принадлежащей всем отрезкам.

Примеры

входные данные

```
2 4
1 2
3 4
```

выходные данные

```
4
```

входные данные

```
4 6
1 3
2 3
4 6
5 6
```

выходные данные

```
5
```

Примечание

В первом тесте из условия Саша никогда не сможет поверить Феде на слово, потому что зная, что даже зная $cnt(x_i)$ для каждой точки из отрезка $[1;4]$, он не может быть уверенным, что Федя на самом не нарисовал один отрезок $[1;4]$.

Во втором тесте из условия Саша может назвать максимум 5 точек, например: $1, 2, 3, 5, 6$. Но уже зная информацию о том, сколько отрезков накрывают каждую из целочисленных точек на отрезке $[1;6]$, Саша может быть уверен в том, что в изначальном рисунке не было точки, принадлежащей всем отрезкам.

944E - Федя не врёт!

Первым делом мы хотели бы для каждой из точек от 1 до m узнать, скольким отрезкам она принадлежит. Это можно делать, например, методом сканирующей прямой за $O(m+n)$, либо деревом отрезков с прибавлением на отрезке за $O(n \cdot log(m))$.

Далее, как легко заметить, набор точек является плохим $\iff$ $cnt(x_i)$ до некоторой позиции не убывают, а затем не возрастают, в противном случае существует тройка индексов $i < j < k$,

для которой верно $cnt(i) > cnt(j), cnt(j) < cnt(k)$, наличие которой показывает, что существуют два непересекающихся отрезка(первый — обладающий самым левым из правых концов среди всех, содержащих i и второй — обладающий самым правым из левых концов среди всех отрезков, содержащих k), из чего следует, что не существует точки, принадлежащей всем отрезкам.

Осталось найди самую длинную такую последовательность x_i. Чтобы её найти, будем перебирать серединный элемент – тот, после которого cnt начинает убывать. Для него нам хотелось бы узнать длину наибольшей неубывающей подпоследовательности, заканчивающейся в нём, а также длину набольшей невозрастающей подпоследовательности, начинающейся с него. Ответ на оба эти вопроса можно получить за $O(1)$, если предпосчитать для каждого элемента массива $cnt[i]$ с помощью классического динамического программирования длину наибольшей неубывающей подпоследовательности исходной последовательности, оканчивающейся в нем, и аналогично для развернутой. Асимптотика подсчёта этой динамики: $O(m \cdot log(m))$

Итоговая сложность решения $O(m \cdot log(m))$ или $O((n+m) \cdot log(m))$ для решения с деревом отрезков.

F. Игра с фишками

ограничение по времени на тест

2 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Рассмотрим следующую игру для двух игроков. Есть одна белая фишка и ненулевое количество черных фишек. Каждая фишка расположена на координатной плоскости в точке с целыми координатами x и y.

Игроки по очереди, начиная с белых, передвигают **все** фишки своего цвета на 1 вверх, вниз, влево или вправо. Черные могут выбирать направление для каждой фишки независимо.

После хода белых белая фишка не может находиться в одной точке с черной фишкой. Других ограничений на расположение фишек нет: нескольким черным фишкам разрешено располагаться в одной точке, после хода черных и изначально белая фишка может находится в одной точке с черной. Если в какой-то момент у белых нет хода, то выиграли черные. Если белые сделали хотя бы 10^{100500} ходов, то они выиграли.

Вам нужно решить следующую задачу. Даны начальные положения всех черных фишек. Гарантируется, что в начале игры все эти положения различны. В скольких местах может находиться белая фишка, чтобы при оптимальной игре выигрывали черные?

Входные данные

Первая строка содержит одно целое число n ($1 \le n \le 10^5$) — количество черных фишек.

В ($i + 1$)-й строке находятся два целых числа x_i, y_i ($- 10^5 \le x_i, y_i, \le 10^5$) — координаты точки, в которой стоит i-я черная фишка изначально.

Гарантируется, что все начальные позиции черных фишек различны.

Выходные данные

Выведите количество точек, в которых может стоять белая фишка в начале игры, чтобы при оптимальной игре обоих игроков выигрывали черные.

Примеры

входные данные

```
4
-2 -1
0 1
0 -3
2 -1
```

выходные данные

```
4
```

входные данные

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

выходные данные

```
2
```

входные данные

```
16
2 1
1 2
-1 1
0 1
0 0
1 1
2 -1
2 0
1 0
-1 -1
1 -1
2 2
0 -1
-1 0
0 2
-1 2
```

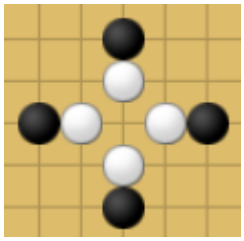
выходные данные

```
4
```

Примечание

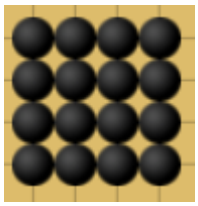
В первом и втором тесте черными кругами обозначены положения черных фишек, белыми кругами — возможные положения белых фишек, при которых выигрывают черные.

Первый тест:



Второй тест:

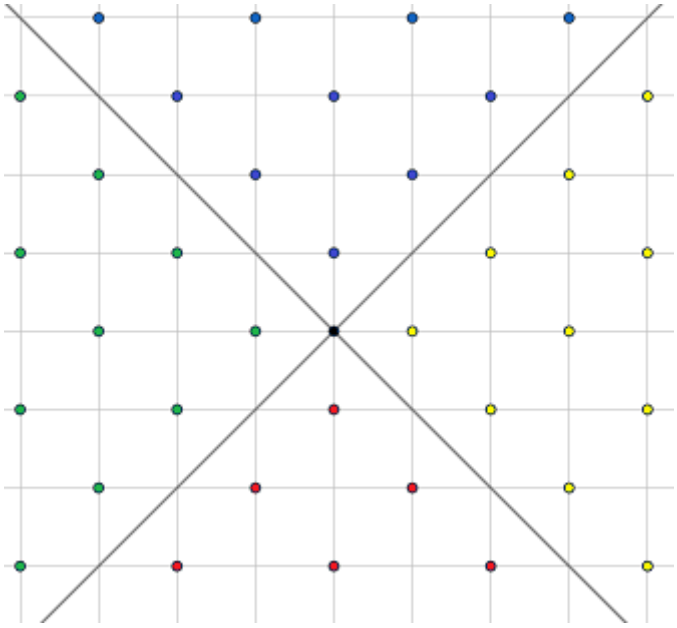
В третьем тесте белые фишки должны располагаться во внутреннем квадрате 2×2, чтобы выиграли черные.



944F - Игра с фишками

Заметим, что если черная и белая фишка находятся в начале игры в точках $\{x, y\}$ с одинаковой четностью $x + y$, то перед ходом белых черная фишка не будет на манхэтеннском расстоянии 1 от белой и не может ее заблокировать. То есть черная фишка с нечетным $x + y$ может хоть как-то повлиять на игру, только если белая фишка имеет четное $x + y$, и наоборот, черная фишка с четным $x + y$ влияет на игру, только если белая фишка имеет нечетное $x + y$. Значит, можно независимо решить задачу для двух наборов черных фишек с одинаковой четностью и сложить ответы. Заметим, что решение для одной четности сводится к решению для другой четности, например, сдвигом всех фишек на 1 вверх. Поэтому дальше будем считать, что для всех черных фишек $x + y$ нечетно. Тогда нас интересуют только положения белой фишки с четным $x + y$.

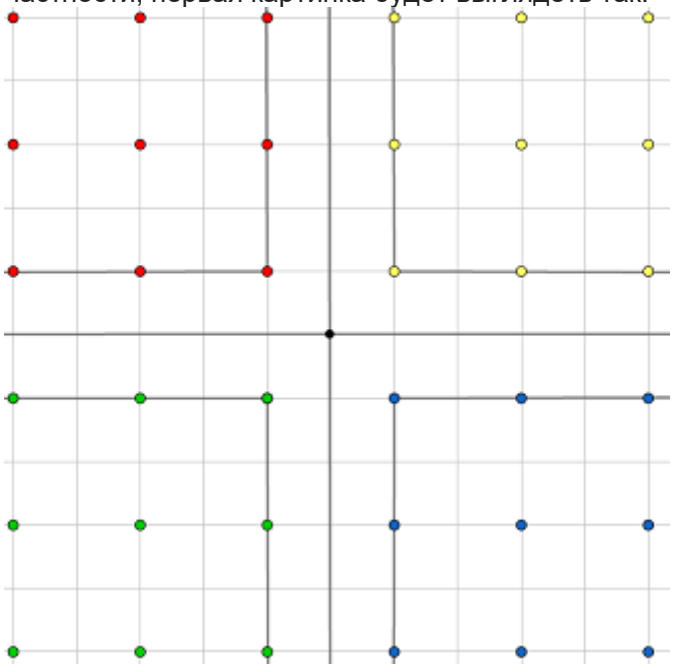
Посмотрим на картинку. Если черная фишка находится в черной точке, то она может помешать белой фишке бесконечно двигаться вверх, вниз, влево, вправо, если белая фишка находится в красных, синих, желтых и зеленых точках, соответственно (то есть какие бы ходы не сделали бы белые, количество ходов вверх, вниз, влево или вправо, соответственно, не может быть бесконечным).



Заметим, что одна черная фишка может остановить белую фишку при движении в максимум одном направлении. Если найдутся четыре черных фишки таких, что каждая может остановить белую фишку в одном из направлений, то черные могут выиграть, используя только эти четыре фишки, и произвольно передвигая остальные. Если найдется направление, при движении в котором, белую фишку никто не может остановить, то белые выиграли.

Следовательно, каждая черная фишка порождает четыре угла разных типов. Если точка $\{x+y\}$находится на пересечении четырех углов разных типов и $x+y$ четно, то такую точку нужно посчитать в ответе.

Заменим каждую точку $\{x, y\}$ на точку $\{x+y, x-y\}$. Тогда четыре типа углов сохранятся, а четными должны быть обе координаты белой фишки, а не только сумма координат. В частности, первая картинка будет выглядеть так:



Оставим только точки с четными координатами и поделим каждую координату на два. По-прежнему каждая черная фишка порождает четыре угла, белая фишка должна находится на пересечение четырех углов разных типов, но больше нет требований четности чего-либо.

Как же находить число точек на пересечении четырех углов разных типов эффективно? Давайте для каждого типа углов и для каждой x-координаты найдем полуинтервал y-координат, что точка, лежащая в данном полуинтервале, покрыта углами текущего типа. Если мы сможем это найти, то нужно просуммировать для каждой x-координаты длину пересечения соответствующих полуинтервалов каждого типа, и это будет ответом.

Рассмотрим углы только одного типа, так как для других типов все делается симметрично. Пусть это углы, стороны которых направлены вверх и вправо. Тогда заметим, что для каждой x-координаты полуинтервал будет вида $[L_x, \infty)$, где L_x - это минимальная y-координата у вершин углов, находящихся не правее x. Следовательно, можно отсортировать все вершины углов по x-координате и написать довольно простой сканлайн.

G. Выставка монет

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

Аркадий и Кирилл пришли на выставку редких монет. Все монеты расположены в один ряд и пронумерованы слева направо от 1до k, причем некоторые монеты лежат вверх аверсом (главной стороной), а некоторые — вверх реверсом (другой стороной).

Аркадий и Кирилл сделали несколько фотографий монет каждый, причем на каждой фотографии изображены верхние стороны нескольких подряд идущих монет. Аркадий интересуется аверсами, поэтому на каждом снимке, который сделал он, есть хотя бы одна монета, лежащая аверсом вверх. Кирилл, наоборот, интересуется реверсами, поэтому на каждом его снимке есть хотя бы одна монета, лежащая вверх реверсом.

Теперь снимки утеряны, и ребята лишь помнят границы отрезков монет, попавших на каждый из снимков. По данной информации о снимках вычислите остаток от деления на $10^9 + 7$ количества способов выбрать для каждой монеты сторону, которой она будет лежать вверх, так, чтобы на каждом снимке Аркадия была бы хотя бы одна монета вверх аверсом, а на каждом снимке Кирилла была хотя бы одна монета вверх реверсом.

Входные данные

Первая строка содержит три целых числа k, n и m ($1 \le k \le 10^9$, $0 \le n, m \le 10^5$) — общее число монет, количество снимков Аркадия и Кирилла, соответственно.

Следующие n строк содержат описания снимков Аркадия, по одному в строке. Каждая из этих строк содержит два целых числа l и r ($1 \le l \le r \le k$), означающие, что среди монет с l-й по r-ю должна быть хотя бы одна вверх аверсом.

Следующие m строк содержат описания снимков Кирилла, по одному в строке. Каждая из этих строк содержит два целых числа l и r ($1 \le l \le r \le k$), означающие, что среди монет с l-й по r-ю должна быть хотя бы одна вверх реверсом.

Выходные данные

Выведите одно целое число — количество способов выбрать для каждой монеты, какой стороной она будет лежать вверх, по модулю $10^9 + 7 = 1000000007$.

Примеры

входные данные

```
5 2 2
1 3
3 5
2 2
4 5
```

выходные данные

```
8
```

входные данные

```
5 3 2
1 3
2 2
3 5
2 2
4 5
```

выходные данные

```
0
```

входные данные

```
60 5 7
1 3
50 60
1 60
30 45
20 40
4 5
6 37
5 18
50 55
22 27
25 31
44 45
```

выходные данные

```
732658600
```

Примечание

В первом примере возможны следующие расположения монет («A» — аверс, «P» — реверс):

- APAAP,
- APAPA,
- APAPP,
- PPAAP,
- PPAPA,
- PPAPP,
- APPAP,

- APPPA.

Во втором примере данная информация противоречива: согласно снимкам, вторая монета должна лежать одновременно аверсом и реверсом вверх, что невозможно. Значит, ответ равен 0.

944G - Выставка монет

Будем обозначать монету, лежащую вверх аверсом, за 0, а монету, лежащую вверх реверсом, за 1. Тогда нам требуется посчитать количество бинарных строк длины k таких, что на n заданных отрезках был бы хотя бы один 0, а на m заданных отрезков — хотя бы одна 1.

Воспользуемся методом динамического программирования. Пусть, для начала, $dp[i][l_0][l_1]$ — количество бинарных строк длины i таких, что последний ноль стоит на позиции l_0, последняя единица — на позиции l_1, и все ограничения на наличие нулей и единиц на заданных отрезках выполнены для всех отрезков, правая граница которых не превосходит i. Переходы построить достаточно просто: переберем, какая цифра будет стоять на позиции $i+1$, обновим l_0 и l_1 в соответствии с этой информацией, пусть новые значения будут l_0' и l_1'. Теперь рассмотрим все данные нам отрезки, заканчивающиеся в позиции $i+1$. Если среди них есть отрезки $[l, r]$, требующие наличия нуля, и при этом $l > l_0'$, или требующие наличия единицы, и при этом $l > l_1'$, то такое продолжение строки нам не подходит. Иначе добавим к ответу $dp[i+1][l_0'][l_1']$ значение $dp[i][l_0][l_1]$. Такое решение работает за $O(k^3 + n + m)$, если заранее для всех величин r сохранить отрезки, заканчивающиеся в позиции r. Такое решение слишком медленное и требует большой объем памяти.

Первое, что можно улучшить в этом решении — заметить, что либо $l_0 = i$, либо $l_1 = i$. Действительно, на позиции i должен стоять либо 0, либо 1. Поэтому вместо предыдущих подзадач достаточно рассмотреть подзадачи $dp_0[i][l_1]$ (при этом подразумевается $l_0 = i$) и $dp_1[i][l_0]$ (здесь $l_1 = i$). Переходы осуществляются аналогично предыдущему решению, асимптотика $O(k^2 + n + m)$, что тоже слишком медленно.

Далее заметим, что все позиции, в которых не заканчивается ни один интересный нам отрезок, можно рассматривать аналогично, так как переходы осуществляются одинаково. В то же время, не имеет значения, находится ли последний 0 в позиции l_0 или в позиции $l_0 + 1$, если в позиции $l_0 + 1$ не начинается ни один отрезок. Аналогично для l_1. Поэтому осуществим операцию, известную как сжатие координат. А именно, найдем все точки x_i такие, что позиции x_i и $x_i + 1$ покрываются различным подмножеством данных нам отрезков. Теперь воспользуемся методом динамического программирования, пусть $dp_0[i][l_1]$ — количество бинарных строк длины x_i таких, что последний символ — 0, последняя единица находится в диапазоне позиций от $x_{l_1 - 1} + 1$ до x_{l_1}, и все ограничения на присутствие единиц и нулей выполнены для всех данных нам отрезков, правая граница которых не превосходит x_i. Аналогично определим $dp_1[i][l_0]$. Рассмотрим возможные переходы в таком методе

динамического программирования, без ограничения общности будем считать, что переход осуществляется из состояния $dp_0[i][l_1]$ в некоторое другое состояние $dp_?[i+1][?]$. Переходы из $dp_1[i][l_0]$ рассматриваются аналогично. В этом случае мы должны дописать бинарную строку длины $(x_{i+1} - x_i)$ к уже существующей. Возможны три случая:

- Все дописываемые символы равны 0, тогда переход осуществляется в состояние $dp_0[i+1][l_1]$ с коэффициентом 1, так как существует единственный способ дописать строку из нулей.
- Все дописываемые символы равны 1, тогда переход осуществляется в состояние $dp_1[i+1][i]$ с коэффициентом 1, так как существует единственный способ дописать строку из единиц, а последний ноль остается на месте x_i.
- Среди дописываемых символов есть как 0, так и 1. Тогда переход осуществляется в состояния $dp_0[i+1][i+1]$ и $dp_1[i+1][i+1]$ с коэффициентами $2^{x_{i+1} - x_i - 1} - 1$, так как существует именно столько способов дописать строку, содержащую и нули и единицы при фиксированном последнем символе. Такой переход возможен, только если $x_{i+1} - x_i > 1$.

Кроме того, при выполнении переходов необходимо рассмотреть все отрезки, заканчивающиеся в x_{i+1} и не учитывать те переходы, которые не удовлетворяют заданным ограничениям. Такое решение работает за $O((n+m)^2 \cdot \log(k))$ и все еще работает слишком медленно. Здесь логарифм появился из-за того, что необходимо быстро вычислять значения 2^q для некоторых целых q, что можно сделать быстрым возведением в степень.

Остался последний шаг для получения полного решения. Заметим, что в динамическом программировании, описанном выше, можно разделить переходы, связанные с дописыванием цифр к строке, и учет ограничений, наложенных данными отрезками. Так, можно сначала выполнить все переходы из $dp_*[i][*]$ в $dp_*[i+1][*]$, не учитывая отрезки, заканчивающиеся в x_{i+1}, а затем учесть эти отрезки, занулив значения $dp_0[i+1][l_1]$, где $l_1 < l$, где $[l, x_i]$ — некоторый отрезок, ставящий ограничение на наличие единицы, и значения $dp_1[i+1][l_0]$, где $l_0 < l$, где $[l, x_i]$ — некоторый отрезок, ставящий ограничение на наличие нуля. Кроме того, заметим, что переходы, имеющие коэффициент, отличный от 1, затрагивают только значения $dp_*[i+1][i]$ и $dp_*[i+1][i+1]$, а значения $dp_*[i+1][j]$ при $j < i$ равны $dp_*[i][j]$ или 0. Поэтому можно не хранить ответы для подзадач с различным первым параметром динамического программирования отдельно, а хранить только два массива $dp_0[l_1]$ и $dp_1[l_0]$, подразумевая параметр i равный текущему значению. Теперь при переходе от i к $i+1$ требуется обновить значения $dp_*[i]$ и $dp_*[i+1]$, а также обнулить некоторый префикс dp_0 и некоторый префикс dp_1 в зависимости от левых концов отрезков, заканчивающихся в x_i. Обновленные значения $dp_*[i]$ и $dp_*[i+1]$ легко вычислить через сумму элементов в этих массивах и величину $x_{i+1} - x_i$. Теперь можно использовать одну из многочисленных структур данных для обнуления на префиксе и подсчета суммы и получить решение, имеющее временную асимптотику $O((n+m)\log(n+m)\log(k))$ и использующее $O(n+m)$ памяти. Такого решение уже достаточно для полного решения задачи.

Кроме того, для упрощения реализации и асимптотики можно заметить, что, так как обнуление всегда происходит на префиксе, который потом не будет обновляться никак иначе, кроме как обнуляться снова, то можно вместо структуры данных поддерживать указатель на первый еще не обнуленный элемент и увеличивать его по мере обнуления. Это также позволяет легко поддерживать текущую сумму всех элементов без использования структур данных. Такое решение имеет временную асимптотику $O((n+m) \cdot (\log(n+m) + \log(k)))$, первый логарифм в асимптотике берется из сортировки, необходимой для сжатия координат. Именно такая идея и реализована в авторском решении.

```
/**
 * This is solution for problem not-equal-segments
 * This is nk_ok.cpp
 *
 * @author: Nikolay Kalinin
 * @date: Thu, 01 Mar 2018 21:12:52 +0300
 */
#include <bits/stdc++.h>


using namespace std;


using ll = long long;
using ld = long double;
using D = double;
using uint = unsigned int;
```

```cpp
template<typename T>
using pair2 = pair<T, T>;

#ifdef WIN32
    #define LLD "%I64d"
#else
    #define LLD "%lld"
#endif

#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second

const int MOD = 1000'000'007;

struct tev
{
    int x, from, type;
```

```
};

inline bool operator<(const tev &a, const tev &b)
{
    return a.x < b.x;
}

vector<tev> events;
vector<pair2<int>> dp0, dp1;
int k, n, m;

inline int powmod(int a, int b)
{
    int tmp = a;
    int ans = 1;
    while (b)
    {
        if (b & 1) ans = ((ll)ans * tmp) % MOD;
        tmp = ((ll)tmp * tmp) % MOD;
        b >>= 1;
```

```
    }
    return ans;
}

inline int get(int x)
{
    return (powmod(2, x) - 1 + MOD) % MOD;
}

int main()
{
    scanf("%d%d%d", &k, &n, &m);
    for (int i = 0; i < n; i++)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        a--, b--;
        events.pb({a - 1, -1, -1});
        events.pb({b, a, 0});
    }
```

```
for (int i = 0; i < m; i++)

{

    int a, b;

    scanf("%d%d", &a, &b);

    a--, b--;

    events.pb({a - 1, -1, -1});

    events.pb({b, a, 1});

}

events.pb({k - 1, -1, -1});

sort(all(events));

int last = -1;

dp0.push_back({-1, 1});

int cur0 = 0;

int sum0 = 1;

dp1.push_back({-1, 0});

int cur1 = 0;

int sum1 = 0;

for (auto t : events)

{

    if (t.x > last)
```

```
{
    dp1.back().se = (dp1.back().se + sum0) % MOD;
    dp0.back().se = (dp0.back().se + sum1) % MOD;
    dp1.pb({t.x, (ll)get(t.x - last - 1) * (sum0 + sum1) % MOD});
    dp0.pb(dp1.back());
    int newsum0 = ((ll)sum0 + sum1 + dp0.back().se) % MOD;
    int newsum1 = ((ll)sum0 + sum1 + dp1.back().se) % MOD;
    sum0 = newsum0;
    sum1 = newsum1;
}
if (t.type == 0)
{
    while (cur1 < (int)dp1.size() && dp1[cur1].fi < t.from)
    {
        sum1 = (sum1 - dp1[cur1].se + MOD) % MOD;
        cur1++;
    }
}
if (t.type == 1)
{
```

```
            while (cur0 < (int)dp0.size() && dp0[cur0].fi < t.from)
            {
                sum0 = (sum0 - dp0[cur0].se + MOD) % MOD;
                cur0++;
            }
        }
        last = t.x;
    }
    cout << (sum0 + sum1) % MOD << endl;
    return 0;
}
```