Gridland Metro

locked

by nabila_ahmed

The city of Gridland is represented as an $n \times m$ matrix where the rows are numbered from 1 to n and the columns are numbered from 1 to m.

Gridland has a network of train tracks that always run in straight horizontal lines along a row. In other words, the start and end points of a train track are (r, c_1) and (r, c_2), where r represents the row number, c_1 represents the starting column, and c_2 represents the ending column of the train track.

The mayor of Gridland is surveying the city to determine the number of locations where lampposts can be placed. A lamppost can be placed in any cell that is *not occupied* by a train track.

Given a map of Gridland and its k train tracks, find and print the number of cells where the mayor can place lampposts.

Note: A train track may (or may not) overlap other train tracks within the same row.

Input Format

The first line contains three space-separated integers describing the respective values of n (the number of rows), m (the number of columns), and k (the number of train tracks).
Each line i of the k subsequent lines contains three space-separated integers describing the respective values of r, c_1, and c_2 that define a train track.

Constraints

- $1 \le n, m \le 10^9$
- $0 \le k \le 1000$
- $1 \le r \le n$
- $1 \le c_1 \le c_2 \le m$

Output Format

Print a single integer denoting the number of cells where the mayor can install lampposts.

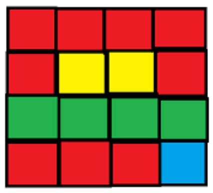
Sample Input

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

Sample Output

```
9
```

Explanation



In the diagram above, the yellow cells denote the first train track, green denotes the second, and blue denotes the third. Lampposts can be placed in any of the nine red cells, so we print **9** as our answer.

Submissions: 3370
Max Score: 25
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)

C#

```
using System;
using System.Collections.Generic;
using System.IO;
class Solution {
     static List<string> intercalarIntervalos(string a, string b)
        {
            string[] a_split = a.Split(' ');
            string [] b_split = b.Split(' ');

            int a_ini = int.Parse(a_split[0]);
            int a_fin = int.Parse(a_split[1]);
            int b_ini = int.Parse(b_split[0]);
            int b_fin = int.Parse(b_split[1]);

            List<string> intervalos = new List<string> ();
            if (a_ini < b_fin && b_ini <= a_fin)
            {
                //intervalos.Add(a_ini + " " + b_fin);
                intervalos.Add(Math.Min(a_ini, b_ini) + " " + Math.Max(a_fin, b_fin));
            }
            else if (b_ini < a_fin && b_fin >= a_ini)
            {
                //intervalos.Add(b_ini + " " + a_fin);
                intervalos.Add(Math.Min(b_ini, a_ini) + " " + Math.Max(b_fin, a_fin));
            }

            return intervalos;
        }


        static void Main(string[] args)
        {

            //-------------------------
            //List<string> lista = new List<string>();

            //using (StreamReader esc = new StreamReader("C:\\test.txt"))
            //{
            //    string line;
            //    while ((line = esc.ReadLine()) != null)
            //    {
            //        //System.Console.WriteLine(line);
            //        lista.Add(line);
            //    }
            //}
            //int indice_lista = 0;


            //-------------------------

            //string[] input1 = "402159386 855281517 951".Split(' ');
            string[] input1 = Console.ReadLine().Split(' ');
            long n = long.Parse(input1[0]);
            long m = long.Parse(input1[1]);
            long k = long.Parse(input1[2]);

            //HashSet<string> filas = new HashSet<string>();
            Dictionary<long, List<string>> filas = new Dictionary<long, List<string>>();
            for (int i = 0; i < k; i++)
            {
                string[] input2 =  Console.ReadLine().Split(' ');
                long r = long.Parse(input2[0]);
                long c1 = long.Parse(input2[1]);
                long c2 = long.Parse(input2[2]);
                string intervalo_actual = c1.ToString() + " " + c2.ToString();

                if (filas.ContainsKey(r))
                {
                    List<string> value = filas[r];
                    foreach (string inter in value)
                    {
                        List<string> mezclados =
                            intercalarIntervalos(intervalo_actual, inter);
                        if (mezclados.Count == 1)
                        {
                            value.Remove(intervalo_actual);
                            value.Remove(inter);

                             List<string> nueva_lista = new List<string>();
                             nueva_lista.AddRange(value);
                             nueva_lista.Add(mezclados[0]);
                             filas[r] = nueva_lista;
                            break;
                        }
                        else if (mezclados.Count == 0)
                        {
                            filas[r].Add(intervalo_actual);
                            break;
                        }
                    }
                }
                else
                {
                    List<string> aux = new List<string>();
                    aux.Add(intervalo_actual);
                    filas[r] = aux;
                }
            }

            long train_track = 0;

            foreach (KeyValuePair<long, List<string>> kvp in filas)
            {
                foreach (string elem in kvp.Value)
                {
                    string[] inter = elem.Split(' ');
                    train_track += long.Parse(inter[1]) - long.Parse(inter[0])+1;
                }
            }

            Console.WriteLine((n * m) - train_track);

            Console.ReadLine();
        }
}
```

Line: 114 Col: 10

Upload Code as File

Test against custom input

Run Code

Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature