Practice Compete Jobs Rank Leaderboard

Dashboard > Algorithms > Graph Theory > Quadrant Queries

Badge Progress (Details)

Points: 126 Rank: 270114

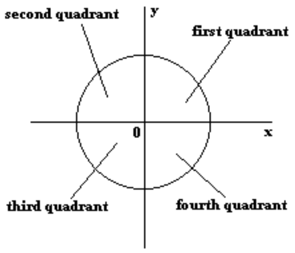
Quadrant Queries

by HackerRank

Problem Submissions Leaderboard Discussions Topics

There are n *2D* points on a plane, and the i^{th} point, p_i, has coordinates (x_i, y_i), where $1 \leq i \leq n$. There are three types of queries:

1. `X i j` Reflect all points in the inclusive range between points p_i and p_j along the x-axis.
2. `Y i j` Reflect all points in the inclusive range between points p_i and p_j along the y-axis.
3. `C i j` Count the number of points in the inclusive range between points p_i and p_j in each of the 4 quadrants. Then print a single line of four space-separated integers describing the respective numbers of points in the first, second, third, and fourth quadrants. Recall that the four quadrants of a graph are labeled as follows:



Given a set of n points (where each point p is indexed from 1 to n) and q queries, perform each query in order.

Input Format

The first line contains a single integer, n, denoting the number of points.
Each line i of the n subsequent lines contains two space-separated integers describing the respective x_i and y_i values for point p_i on the *2D* plane.
The next line contains a single integer, q, denoting the number of queries.
Each of the q subsequent lines contains three space-separated values describing a query in one of the three forms defined above. You must process each query in the same order as it's read from stdin.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq q \leq 10^6$
- It is guaranteed that no point lies on the x or y axes.
- All (x_i, y_i) points will fit in a 32-bit signed integer.
- In all queries, $1 \leq i \leq j \leq n$.

Output Format

For each query of type `C i j`, print four space-separated integers describing the number of points having indices in the inclusive range between points p_i and p_j in the respective first, second, third, and fourth graph quadrants.

Sample Input

```
4
1 1
-1 1
-1 -1
1 -1
5
C 1 4
X 2 4
C 3 4
Y 1 2
C 1 3
```

Sample Output

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

Explanation

Query `C 1 4` asks you to consider the set of points having indices in $\{1, 2, 3, 4\}$, meaning $p_1 = (1, 1)$, $p_2 = (-1, 1)$, $p_3 = (-1, -1)$, and $p_4 = (1, -1)$; amongst those points, how many of them lie in the respective first, second, third, and fourth quadrants? Because we have one point in each quadrant, we print `1 1 1 1` on a new line.

Recall that queries in the form `X i j` and `Y i j` are telling us to take all the points in the inclusive range between indices i and j (i.e., points p_i and p_j) and reflect them along the axis specified by the first character of the query. Note that i and j here refer to actual point numbers/subscripts and *not* coordinates on the plane.

So when we process query `X 2 4`, we reflect the points in the inclusive range between indices 2 and 4 (i.e., points p_2, p_3, and p_4) along the x-axis. This means our coordinates are now $p_1 = (1, 1)$, $p_2 = (-1, -1)$, $p_3 = (-1, 1)$, and $p_4 = (1, 1)$.

Next, `C 3 4` tells us to consider the set of points in the inclusive range between indices 3 and 4 (i.e., points p_3 and p_4) and print the number of points in this range falling on each respective axis. Point $p_3 = (-1, 1)$ lies in quadrant 2 and point $p_4 = (1, 1)$ lies in quadrant 1, so we print `1 1 0 0` on a new line.

Next, `Y 1 2` tells us to reflect the points in the inclusive range between indices 1 and 2 (i.e., points p_1 and p_2) along the y-axis. This means our coordinates are now $p_1 = (-1, 1)$, $p_2 = (1, -1)$, $p_3 = (-1, 1)$, and $p_4 = (1, 1)$.

Finally, `C 1 3` tells us to count the number of points in each quadrant that fall in the inclusive range between indices 1 and 3 (i.e., points p_1, p_2, and p_3). Point $p_1 = (-1, 1)$ is in quadrant 2, point $p_2 = (1, -1)$ is in quadrant 4, and point $p_3 = (-1, 1)$ is in quadrant 2. Thus, we print `0 2 0 1` on a new line.

Submissions:2262

Max Score:100

Difficulty: Advanced

Rate This Challenge:

☆ ☆ ☆ ☆ ☆

Need Help?

Segment Tree

More

Current Buffer (saved locally, editable)

Java 7

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

```

```
public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
    }
}
```

Line: 1 Col: 1

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