DOWNLOAD - Editorial

PROBLEM LINKS

8

Practice
Contest

DIFFICULTY

1

MEDIUM

PREREQUISITES

Binary Index Tree , sorting , coordinate compression

PROBLEM

The problem statement is simple. We are given N intervals on 1D axis and asked Q queries. Each query is a small set of K points and we are asked to find the number of distinct intervals that cover at least one of the given K points.

EXPLANATION

This problem can be solved by handling queries online, using RangeTree or other similar data structures to count the number of points in a 2-side bounded box in two-dimensions. But in contests, its faster to implement an offline approach and so we will explain an offline method here. If you are not familiar with the terms, offline means, if we know all the queries before hand. Online means, we just have the input data and the queries are given one by one.

We will use the standard Binary Index Tree (BIT) here. Using BIT means, we play a lot with its indices, mapping the input data to indices. The constraints on the input values are huge (10^9) but we do not need the absolute values of the input. Only their relative order is important to us. So the first step is to compress the input data such that we only need O(n) distinct values to represent the input. For eg. if we have the array { 10, 24, 1000, 3, 30 , 24 } , it is same as having { 2 , 3 , 5 , 1 , 4 , 3 } as the relative order between all pairs of input values is still preserved in the new transformed array. This can be done by sorting the array and finding the position of each value after removing duplicates.

Given points X1 < X2 < ... < Xk, we need to find the number of distinct intervals [Si, Ei] that cover at least one of the given K points. If we sum up the number of intervals covered by each point, we may count some intervals many times. If Pi is the set of intervals covered by point Xi , we want to find the number of intervals in the set P1 U P2 U ... U Pk. The first method that strikes us is using inclusion-exclusion. For that we need to answer the following question : Given a set of points, find the number of intervals, each covering ALL the points. Note that to answer this, its sufficient to find the intervals that cover the extreme points in this set. All other points are anyway lying in between the extremes and will be covered. This will reduce the need for going through all $O(2^k)$ subsets to considering only all possible pairs of extremes, $O(k^2)$

Notice how many times the count for a pair of extreme points is included. Fix a pair of extreme points Xi and Xj and find the count of number of intervals, each covering both the points Xi and Xj. This count is added for all sets with odd number of elements and subtracted for all sets with even number of elements. So, for a given pair of extreme points, its count is nullified if there is at least one other point present in between them.

So the final answer is to sum up the number of intervals included by each point and subtracting from it, number of intervals included by each of the (k-1) pairs of adjacent points. This can be solved offline using BIT. We collect all the query points, beginning of intervals Si and end of intervals Ei and sort them and process in non-decreasing order. The following cases will be encountered.

1. Beginning of interval Si : Increment BIT[Si] by 1
2. End of interval Ei : Decrement BIT[Si] by 1 , the Si corresponding to the end Ei
3. Query point Xi : QueryBIT(Xi) gives the number of active intervals that include Xi. Add it to the answer corresponding to this query. QueryBIT(X(i-1)), querying at the just preceding point of Xi in its query point set, gives the number of intervals that include both X(i-1) and Xi. Subtract it from the answer corresponding to this query.

This offline method takes in total O(N logN + Q * K * log N) time.

SETTER'S SOLUTION

Can be found here

APPROACH:

The setter's solutions uses the approach mentioned in the Explanation above.

Tags:

editorial **×3,963**
medium **×707**
sorting **×213**
bit **×195**
cook23 **×8**

Asked: **18 Jun '12, 00:36**
Seen: **5,777 times**
Last updated: **14 Sep, 21:22**

Related questions

- CLOST - Editorial
- PPLUCKY - Editorial
- SEABAL - Editorial
- CHEFD - Editorial
- HOB - Editorial
- TARPRACT - Editorial
- LUCKYSWP - Editorial
- POSTAL - Editorial
- TOOLS - Editorial
- NEWREST - Editorial

TESTER'S SOLUTION

Can be found here

APPROACH:

The tester's solution is also offline processing, but uses a different approach. Instead of finding the number of intervals including the point set, we find the number of intervals that do not overlap with any of the points in the query set. For a given point set X1 < X2 < ... < Xk , consider X0 = 0 and X(k+1) = infinity. For each pair Xi and Xi+1 for i = 0 to k , we find the number of intervals starting after Xi and ending before Xi+1 and subtract this from the total number of intervals N. We can use BIT here too.

We will update this section later this week with more details on tester's approach. You are free to edit this part and contribute to the editorials.

bit sorting medium editorial cook23

This question is marked "community wiki".

edited **25 Dec '12, 14:11**
admin ♦♦
11.4k●346●472●491

asked **18 Jun '12, 00:36**

flying_ant ♦♦
1●7●13●13
accept rate: 0%

I think you are describing the tester's solution.

Infinity (24 Sep '14, 13:10)

5 Answers: oldest newest **most voted**

1

The O(k^2) part confused a little, in the end only 2k - 1 pairs are used.

link | award points

edited **18 Jun '12, 10:48**

answered **18 Jun '12, 10:48**

MarioYC
222●2●2●6
accept rate: 0%

0

If the Intervals were like [1,10] [2,5] [7,9] and x1=3 , x2=8

So according to "Query point Xi" -> bit(x1)=2 and bit(x2)=2 , thus it means that none of the intervals are covering both x1 and x2 , which is not true (since [1,10] does cover both x1 and x2)

reply asap

link | award points

answered **18 Mar '13, 03:21**
crazygeek
1●1
accept rate: 0%

0

Thank you.

link | award points

answered **03 Apr '13, 22:16**
jpbeauty
1
accept rate: 0%

0

you can refer lecture 6 and 7 for more clearance https://www.youtube.com/channel/UCyZtjmvybLLIk2KZgLJL6ZA

link | award points

answered **27 Jun '14, 16:18**

chomu1850
30●5
accept rate: 0%

0

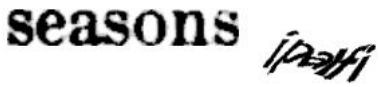
shouldnt the complexity be N logN + Q * K^2 * log N cause we need to go over all pairs?

link | award points

answered **14 Sep, 21:22**
inonkp
1
accept rate: 0%

Your answer

[hide preview] community wiki



Type the text

Privacy & Terms

Post Your Answer