CODE FORCES
Sponsored by Telegram

Enter | Register

HOME TOP CONTESTS GYM PROBLEMSET GROUPS RATING EDU API CALENDAR HELP 10 YEARS!

Please, try EDU on Codeforces! New educational section with videos, subtitles, texts, and problems. ×

KARTIK8800 BLOG TEAMS SUBMISSIONS GROUPS CONTESTS

kartik8800's blog

CSES Range Queries section editorial

By **kartik8800**, 2 months ago,

Hello Codeforces, In this blog I will try to write a well detailed editorial for the CSES Range Queries section. The motivation for this editorial comes from https://codeforces.com/blog/entry/70018.

Quoting **icecuber** "I think CSES is a nice collection of important CP problems, and would like it to have editorials. Without editorials users will get stuck on problems, and give up without learning the solution. I think this slows down learning significantly compared to solving problems with editorials. Therefore, I encourage others who want to contribute, to write editorials for other sections of CSES."

So here I am writing an editorial for the range queries section.

If you find any error or maybe have a better solution to some problem please do share.

Range Sum Queries I

Given array is A[1..N], for each query of form (L,R) we need to output A[L] + A[L+1] + ... + A[R].
Define an array Prefix such that **Prefix[i] = A[1] + A[2] + .. + A[i].**
Prefix[R] = A[1] + A[2] + ... + A[R] and Prefix[L-1] = A[1] + A[2] + ... + A[L-1].
Consider Prefix[R] — Prefix[L-1] = (A[1] + A[2] + ... + A[R]) — (A[1] + A[2] + ... + A[L-1]) = (A[L] + A[L+1] + ... + A[R]).

So for every query simply output **Prefix[R] — Prefix[L-1]**.

How Do I build the prefix Array.

```
prefix[0] = 0
for i = 1 to N
     prefix[i] = ar[i] + prefix[i-1]
```

Time complexity : O(N) to build prefix array and O(1) per query.

Range Minimum Queries I

Two possible ways are as follows :
1. Build a Range minimum query segment tree in O(N) time and answer each query in O(logN).
2. Build a sparse table in O(NlogN) time and answer each query in O(1).

Code for approach 1

```
int small(int x,int y){return min(x,y);}
SegmentTree < int > rangeMinQueries(dataVector,INT_MAX,small);
For answering a query : rangeMinQueries.query(l,r)
```

Refer https://codeforces.com/blog/entry/71101 for my segment tree template.

Approach 2 : https://cp-algorithms.com/data_structures/sparse-table.html

Range Sum Queries II

→ Pay attention

Before contest
Codeforces Round #656 (Div. 1)
2 days

Before contest
Codeforces Round #656 (Div. 2)
2 days

Like 86 people like this. Sign Up to see what your friends like.

→ Top rated

#	User	Rating
1	MiFaFaOvO	3681
2	tourist	3669
3	Um_nik	3535
4	300iq	3317
5	ecnerwala	3294
6	maroonrk	3268
7	TLE	3223
8	scott_wu	3209
9	WZYYN	3180
10	boboniu	3174

Countries | Cities | Organizations View all →

→ Top contributors

#	User	Contrib.
1	Errichto	197
2	antontrygubO_o	191
3	pikmike	185
4	Ashishgup	182
5	vovuh	179
6	Um_nik	175
7	Radewoosh	173
8	SecondThread	168
9	Monogon	163
10	McDic	162

View all →

→ Find user

Handle:

So this one is a direct use of segment tree with point updates and I shall use my segtree template to answer this problem.

Code

```
vector dataVector(n);
ll sum(ll x,ll y){return x+y;}
SegmentTree < ll > rangeSumQueries(dataVector,0,sum);
For query of type 1 : rangeSumQueries.update(idx,newVal);
For query of type 2 : rangeSumQueries.query(a,b);
```

Both of these queries can be performed in logN time.
Overall time complexity is O(N) for building segtree and QlogN for Q queries.

Range Minimum Queries II

Again a straightforward segment tree problem and I will use a similar code as I used for the previous problem. This is the same as RMQ1 except that here we also have point updates. Segment tree solution for RMQ1 and RMQ2 will be identical.

Code

```
vector dataVector(n);
int small(int x,int y){return min(x,y);}
SegmentTree < int > rangeMinQueries(dataVector,INT_MAX,small);
For query of type 1 : rangeMinQueries.update(idx,newVal);
For query of type 2 : rangeMinQueries.query(a,b);
```

Both of these queries can be performed in logN time.
Overall time complexity is O(N) for building segtree and QlogN for Q queries.

Range Xor Queries

For this problem we can maintain a segment tree where each node of the tree will store the xor-sum of the range of elements for which it is responsible.
So root of the tree stores : A[1]^A[2]^....^A[N].

To calculate the answer for a particular interior node of the tree we do :
NODE_VAL = LEFT_CHILD_VAL ^ RIGHT_CHILD_VAL
For leaf nodes :
NODE_VAL = A[x], where [x,x] is the range this leaf node is responsible for and if x > N then NODE_VAL = 0 as 0 is the identity for xor_sum.

Again with these observations we can use the same segtree template as follows :

Code

```
vector< int > dataVector(n);
int xor_sum(int x,int y) { return x^y; }
SegmentTree< int > xorSumQueries(dataVector, 0, xor_sum);
For answering the query : xorSumQueries.query(l,r)
```

AC code : https://ideone.com/mPhqas

Range Update Queries

Nice question which can be directly solved with a segment tree with lazy propagation but that is an overkill plus my segtree library does not support lazy propagation as of now.

Let's define a few terms:
SUM[i] = overall update done on the ith element till now
Initially SUM[i] = 0 for all i as no updates have yet been performed, now we would like to track the updates happening so that our answer to a query **2 k** can easily be **v[k] + SUM[k]** where **v** is the initial array.

How to efficiently maintain the **SUM** array? Let us build a range sum query segment tree on some array **X** which has all elements initialized to 0.

Find

→ Recent actions

Um_nik → Support Anton Trygub

prophet_ → Foundation for fixing various CF tools broken by recent security update

prophet_ → Critical Bug in Codeforces: Broken AES encryption results in 403 Forbidden Error. (Solution)

xalanq → Codeforces command-line interface tool

Traviswr → Need help in a bracket sequence problem.

vovuh → Codeforces Round #653 (Div. 3) Editorial

Google_Kickstart_Plag → Is Google Kickstart Credible Anymore?

Ripatti → Solutions for Codeforces Beta Round #65 (Div. 2)

de_dust2 → Google kickstart Round D problem 4 - Locked Doors

karthikeyan_01 → CSES Labyrinth Problem

elena → How to add contest to Gym from Polygon

sarthakmanna → Invitation to Coders' Legacy 2020 (Rated for all)

adyyy → HELP WITH GRUNDY

MikeMirzayanov → Educational Codeforces Round 91 is ruined and unrated

agrawal117 → How to find sum of values in range which are less than 'K' ?

pikmike → Educational Codeforces Round 91 Editorial

MikeMirzayanov → Frequently Asked Questions

pllk → Binary search implementation

KAN → Analysis of Codeforces Round #464

antontrygubO_o → Some thoughts on recent discussions

vovuh → Codeforces Round #506 (Div. 3) Editorial

himanshujaju → 0-1 BFS [Tutorial]

M.Mahdi → Codeforces Round #360 Editorial [+ Challenges!]

Bajrang_Pandey → Whats up with codeforces?

MagentaCobra → Codeforces Round #655 Editorial

Detailed →

DEFINE : RSQ[i] = X[1] + X[2] + .. + X[i] = rangeSumQuery(1,i)

Now say we have a query which tells us to add **u** to all elements in the range [l,r] then if I perform a point update and make X[l] = X[l] + **u** think what happens to RSQ[i] for different values of i.

RSQ[i] is unaffected for all i where i < l
and RSQ[i] = RSQ[i] + **u** for all i >= l.

Effectively we just did a range update on the abstract array **RSQ** and the range update is equivalent to adding **u** to all elements of array **RSQ** from l to N.

But we wanted the range update to be only for the range [l,r], so we should now do a range update in which we subtract **u** from [r+1,N] and this is the same as doing a point update to X[r+1] such that:

X[r+1] = X[r+1] — u

it must be easy to see the abstract array **RSQ** is nothing but the required **SUM** array.

So here is the algorithm :

```
For every range update query (l,r,u):
  point_update(l,current_val + u)
  point_update(r+1,current_val – u)
For every query -> value at pos idx:
  print SUM[idx] + V[idx]
```

AC code : https://ideone.com/vBZpYx
Time complexity per query is logN.

Forest Queries

For every query of the form (x1,y1,x2,y2) we need to answer the number of trees inside the rectangle described by the top left cell of rectangle (x1,y1) and the bottom right cell of rectangle (x2,y2).

Define : DP[i][j] as the number of trees in the rectangle described by (1,1) and (i,j).

Can we use DP matrix to evaluate answers for every query?
Spoiler

YES.

Ok, but how?
Spoiler

first please check this image : https://www.techiedelight.com/wp-content/uploads/Result.png
Trees in rectangle ((x1,y1),(x2,y2)) can be decomposed into the following rectangles :
1. Trees in rectangle ((1,1),(x2,y2)) = DP[x2][y2]
2. Trees in rectangle ((1,1),(x1-1,y2)) = DP[x1-1][y2]
3. Trees in rectangle ((1,1),(x2,y1-1)) = DP[x2][y1-1]
4. Trees in rectangle ((1,1),(x1-1,y1-1)) = DP[x1-1][y1-1]

Finally we have the answer to our query as DP[x2][y2] — DP[x1-1][y2] — DP[x2][y1-1] + DP[x1-1][y1-1]

How to build DP matrix?
Let tree[i][j] = 1, if there is a tree at cell (i,j) else tree[i][j] = 0.

```
DP[0][0] = DP[-1][0] = DP[0][-1] = 0
for i from 1 to N:
   for j from 1 to N:
    DP[i][j] = DP[i-1][j] + DP[i][j-1] – DP[i-1][j-1] + tree[i][j]
```

Time complexity for build O(N*N) and time complexity per query is O(1).
AC code : https://ideone.com/5dGTfY

Hotel Queries

Observation : For each group, we need to find the 1st hotel with enough vacancies and book rooms for the group in that hotel.

Brute force : Start checking every hotel from left to right for the number of rooms it has. As soon as you find a hotel with enough rooms use required number of rooms in this hotel. Repeatedly do this till all groups have been assigned a hotel.

How to optimize?

Try thinking about the following problem instead, is there any hotel in the first **x** hotels which can be assigned to the current group?

Let us say we are able to answer this problem efficiently with a **yes/no**.
Then if the answer is yes, our problem reduces to finding the 1st hotel with enough rooms in the range [1,x] instead of the range [1,N].
if the answer is no, our problem reduces to finding the 1st hotel with enough rooms in the range [x+1,N] instead of the range [1,N].

So clearly, we can simply do a binary search to find the correct hotel and assign it to current group.

How do I know if there is any hotel in the first **x** hotels which can be assigned to the current group?

Spoiler

Let current group size be G. Then if there exists a hotel in the first x hotels with vacancy >= G then our answer is YES else our answer is NO.

if (RANGE_MAX_QUERY(1,X) >= G) return YES else return NO.

Algorithm :

```
For each group gi with size si:
  Find the 1st hotel x such that vacancy(x) >= si.
   Do point update : vacancy(x) = vacancy(x) – si.
   Print x.
  If no valid x exists print 0.
```

Time complexity is O(Mlog^2N), logN steps for the binarySearch and each step of the binary search uses the Range max query segment tree which works in logN time.

List Removals

Brute force is quite simple if you simply simulate what is mentioned in the problem.
Let us try to optimize. So whenever we are asked to delete some **x**th element of the list we need to first locate the **x**th element, print it and then delete it.

How can we make the above processes faster?
Let us keep a boolean array **PRESENT** of size N and PRESENT[i] = 1 if the ith element of the list has not yet been deleted, 0 otherwise.

Now let us say we have the query : delete the **x**th element of the list, then this means we are going to delete the element at the **j**th index of the initial list such that :

1. PRESENT[j] = 1.
2. sum of PRESENT[i] for all i from 1 to j = x.

Why above conditions are necessary and sufficient to locate the correct element?
If are we deleting the element it has to be present in the list currently and so PRESENT[j] should be 1.
If this element at index j(of the initial list) is the **x**th element of the list(current state of the list)

then there are exactly x elements present in the list in range [1,j](of the initial list) and remaining j — x elements got deleted in some previous queries.

How do I find this j?

Spoiler

Binary Search.

Ok, elaborate.

Spoiler

The correct j lies in the range [1,N]. Is the index **mid** the correct index?
Consider following facts :
If number of elements not yet deleted in [1,mid] > x then search range should be updated to [1,mid-1] else If number of elements not yet deleted in [1,mid] < x search range should be updated to [mid+1,N].

else If number of elements not yet deleted from [1,mid] = x then if PRESENT[mid] = 1, mid is the correct index else update search range to [1,mid-1].

How do I find number of elements not yet deleted in the range [1,j]?

Hint : problem comes from range query section

Build a range sum query segment tree on the PRESENT array and find the sum of the range [1,j] in logN time.

Once you have found the correct j, you need to print it and also mark PRESENT[j] = 0 and make the required point update in the segment tree.

Time complexity analysis is similar to previous problem.

AC code : https://ideone.com/anpuXy

Salary Queries

Okay so, this seems a bit hard. Maybe if the max possible salary of the employees was limited to some smaller amount(instead of a billion) we might be able to solve it.

So try solving the problem under the constraint that p,a,b <= 10^7.
Now the problem is much easier if I maintain the number of people with a given salary, let us define
freq[i] : number of employees with the salary i
We may now build a range sum query segment tree on this array and to answer a query we simply calculate the sum of the range [a,b].

For updating the salary of some employee from x to y, we do the point updates freq[x] -= 1 and freq[x] += 1 because now 1 less employee has salary x and 1 more employee has the salary y.

But the problem is not solved, since we needed to do this for max possible salary = 1billion, but now we know how to do it for 10^7.

observation

10^9 = 10^7 * 10^2

So lets group the salaries into 10^7 buckets and each bucket represent a range of 100 different contiguous salary values. The 0th bucket represents salaries from 1 to 100 and ith bucket represents the salaries from (i)*100 + 1 to (i+1)*100.

These buckets will store aggregated number of employees that have salaries in the given range represented by the bucket.

Now for query [a,b] : all the buckets that are entirely in the range [a,b] their aggregate values should be taken and summed up and for the 2 partial buckets(or 1) not entirely included in the range [a,b] we shall do a brute force.

So build a segment tree over the buckets and calculate the sum over all completely included buckets in the range [a,b]. For remaining partially included buckets do a brute force(actually iterate over approx 100 possible values and choose to include those which are required by a particular query, refer code).

A code will make this explanation much more clear.

AC code : https://ideone.com/zg97c8

Time Complexity?

proportional to 10^7 operations to build the tree, per range sum query and per point update operations proportional to log(10^7). Accessing the map is proportional to logN operations, for left extreme and right extreme buckets we will need to do proportional to (logN + 100) operations.
Somewhat proportional to (10^7 + Qlog(10^7) + 200*Q + Q*logN) operations.

other way to do it is using a dynamic segment tree in which you only build a node of the tree when it is needed.

Distinct Values Queries

This is a direct application of the MO's Algorithm. You may read more about MO's algorithm on https://blog.anudeep2011.com/mos-algorithm/

The brute force can be done by simply iterating from index a to b and maintaining number of distinct elements seen till now and a frequency array to indicate which elements and how many occurrences of those elements is present.

Frequency[i] = count of occurences of i in the current range.

Next we try to build the required ADD and REMOVE functions which help MO's algorithm to function properly.
To ADD a new element in the current range simply check if this element is already present(frequency > 0) and if it is present just increase its frequency else if its frequency was 0 then make it 1 and also increase the number of unique elements in the range.
To REMOVE an element from the current range, decrement its frequency by 1, if its frequency reaches 0 then decrease the number of distinct elements in the current range.

After this sort the queries as described by MO's algorithm and you are done.

Twist : We cannot use frequency array as value of individual element can go upto 10^9. So what I'll simply use an unordered_map?

No, unordered_map solution will time out due to high constant factor involved.

How to continue using the frequency array?

Coordinate compression, simply map larger values to smaller values in the range [1,N] and you are done as N can be atmost 2*10^5.

We can do this because we don't really care about the actual values but only about the number of unique values in a range.

Time complexity : O((N+Q)root(N))

AC code : https://ideone.com/RkC547

Subarray Sum Queries

Let us try to keep track of the max sum subarray in a particular range [L,R]. If we were to build a segment tree in which each node of the tree stores max sum subarray of the range that the node is responsible for then the root keeps track of max sum subarray in the range [1,N]. However for segment trees to be efficient we need to generate the answer of interior nodes of the tree using the answers/information provided by the child nodes.

Now let's try to generate the answer for some interior node **P** of the segment tree assuming that we already have the answers for the children of the node **P**.

Node **P** is responsible for the range [l,r], its left child is responsible for the range [l,mid] and its right child is responsible for the range [mid+1,r].

Now we need to find the sum of max sum subarray in the range [l,r].
Assume you have all necessary information about the child nodes but if you have some information about a child node you also need to generate that piece of information for the parent node as well(since this node also has a parent which will use information given by **P** to generate it's answer).

How to relate the answer for parent to the answers about children?

Consider the following possibilities regarding max sum subarray in range [l,r]:
1. It might lie entirely in the range [l,mid] or entirely in range [mid+1,r].
2. It might be some subarray [p,q] such p <= mid and q > mid

Agreed, now what info to keep for every node?

We need to cover all the possibilities discussed above.
If subarray lies entirely in [l,mid] then my subarray for [l,r] is the same as that for [l,mid] and so is the sum, so it is required to know the sum of max sum subarray of range [l,mid] and on similar lines we also need to know sum of max sum subarray of range [mid+1,r].
Now try and think what property is useful to calculate the maximum sum subarray [p,q] such p <= mid and q > mid.
The subarray [p,q] is the union of the subarrays [p,mid] and [mid+1,q]. Among all possible [p,mid] subarrays we should choose p(l<=p<=mid) such that sum of elements in range [p,mid] is maximized and similarly among all possible values of q(mid+1<=q<=r) we should choose q such that sum of elements in range [mid+1,q] is maximized, maximizing these two individually is equivalent to maximizing [p,q].
Call [p,mid] as the suffix and [mid+1,q] as prefix. Our max sum subarray [p,q] such that p <= mid and q > mid is nothing but (suffix_of_left_child) **UNION** (prefix_of_right_child).

So to summarize, for every node which represents the range [l,r] we should store :
1. sum of max sum subarray in the range [l,r].
2. maximum possible sum of some prefix [l,x] (such value of x is chosen, such that l <= x <= r and sum of elements in range [l,x] is maximum possible.)
3. maximum possible sum of some suffix [x,r].

We now know how to calculate sum of max sum subarray for some node using the above mentioned information about children nodes but as discussed we should also calculate prefix and suffix info for the parent also.

How to calculate prefix and suffix for parent node

The maximum sum prefix could be the same as prefix_of_left_child or it could be the entire left_child_range **UNION** prefix_of_right_child.

But now we need the sum of elements of left_child_range as well, which is the sum of all elements in [l,mid]. So this is another information we should store for every node.

Similarly the maximum sum suffix could be the same as suffix_of_right_child or it could be the entire right_child_range **UNION** suffix_of_left_child.

Refer the combine function in the code for more clarity.
AC code : https://ideone.com/MhVmBs

Time complexity : logN per update.

Forest Queries II

Problem is almost the same as Forest Queries but we also have point updates.
I will discuss two approaches, one of them is quite efficient while the other one struggles to run in the time limit(passes if constant factor is low).

Approach 1

Let us do build the same DP matrix which we had built for the problem Forest Queries. If somehow we are able to keep our DP matrix consistent with the updates then our answer will be the same as before.

ANSWER TO QUERY (x1,y1,x2,y2) : DP[x2][y2] — DP[x1-1][y2] — DP[x2][y1-1] + DP[x1-1][y1-1]

How to efficiently track the updates that happen on some entry DP[i][j]?

Which entries of the DP matrix change if cell (i,j) is updated?

DP[x][y] represents the number of trees in the rectangle described by ((1,1),(x,y)).
Only those entries will be affected which contain the cell (i,j). If the rectangle described by ((1,1),(x,y)) contains the cell (i,j) then we have the following conclusions about x and y :
1. x >= i
2. y >= j

What will be the change in these entries?
Either a +1 or a -1, depending upon whether a tree was planted or removed in cell (i,j).

Alright, so now I need to efficiently add or subtract 1 from all the matrix entries (x,y) where that x >= i and y >= j.

So how do we track these updates?

Range query Data structures.
We needed to either add or subtract 1 from all entries (x,y).
Now consider the row X(X >= x) : for this row we need to update entries (X,j), (X, j+1) ... (X,N)
This is nothing but a range update to the range [j,N].
So here is what we can do, keep a fenwick tree for each row. Fenwick[k] represents the fenwick tree for the kth row and Fenwick[k].rangeSum(0,j) represents the net change for the entry (k,j).

For updates, simply do the point updates Fenwick[k].update(j, delta) for all k from i to N where delta is either 1 or -1.

Time complexity O(Q*N*logN)
AC code : https://ideone.com/mcAdwL
Code for fenwick tree is taken from : https://cp-algorithms.com/data_structures/fenwick.html

Approach 2
This approach is more efficient and some might also find it easier to understand.
It uses a 2D Binary Indexed tree to achieve an overall time complexity of O(N^2 + Q*log^2(N)).

You can read more about it here : TopCoder-2DBIT

Range Updates and Sums

No surprises here. My solution will use the data structure and the technique related to that data structure which most would have already guessed after reading the problem statement. The important thing would be a thorough understanding of the concept and a neat implementation(I have tried to make it readable).

what data structure, what technique?

We will use a segment tree with lazy propagation. If you have an alternate approach, please share.

Alright so our segment tree needs to support the following operations :
1. increase values in range [l,r] by x.
2. set values in range [l,r] = x.
3. return sum of values in range [l,r].

Think about what information should we store per node of the tree, so that we are able to lazily update our nodes when a new update comes in and we are able to propagate the updates downward when needed.

To better understand lazy propagation, I recommend reading this : Super amazing theory in 1 comment (My implementation uses applyAggr and compose functions mentioned here).

What info to store per node?

Ofcourse both the updates are needed to be stored along with the sum, what do they represent?

Let's keep the following variables per node:
1. setAll
2. increment
3. Sum : The sum of the values in range [l,r](the range represented by the node)

Representation : All the values in range [l,r] should be set to the value "setAll" and then their values should be increased by "increment". Initially, we may set increment to 0 but then what about setAll? if we set setAll to 0 it would mean all values are 0. So we need an indicator that tells if the value given by setAll is valid or invalid.

So let us also keep another variable setAllValid per node which is 1 if the value given by setAll is valid otherwise 0.

How do I find the actual Sum using these variables?

if setAll is valid, finding sum is quite easy. If L is the number of elements in the subarray which this node represents then our sum will simply be L * (setAll + increment).
if setAll is invalid, then whatever is the value of the sum variable we should add (L * increment) to it.

How do i propagate these updates downward?

First apply them to the node and calculate the sum for this node.
Next push the updates down to the children.
Now the child nodes may have updates of their own as well, how to combine these updates?
Updates propagated down are always more recent.
So if parent node had setAllValid = 1, then all updates in child node are overwritten by the parent update values otherwise the increment of parent and increment of child node summed up make the new increment of the child node and rest everything stays same.

Also after you have pushed the updates down, you may reset the updates of the parent back to default values(increment = 0, setAllValid = 0)

Time Complexity is logN per query.
If something is not explained or if something isn't clear feel free to ask but I recommend understanding lazy prop well before attempting the problem or reading the editorial.
AC code : https://ideone.com/8HQxMk

Please feel free to point out mistakes, suggest better/alternate solutions and to contribute.
I'd be glad to know if this helps :)

P.S. Will add remaining problems in a few days.

UPD : Editorial is almost complete with 2 problems left. 2nd last uses segment tree with lazy prop and I am guessing the last one uses some kind of persistent data structure, will add soon.

+181 kartik8800 2 months ago 51

Comments (51)

Write comment?

Gaurav678

2 months ago, # | 0

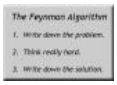
nice one thanks.

if possible , can you please provide all the other links for cses editorials like these 2 by you and icecuber
→ Reply

kartik8800

2 months ago, # ^ | 0

I am unaware of any other editorial for CSES sections, if you or anyone knows about some section which is available share it and I'll add the link to them in this blog.
→ Reply



Colossus20

2 months ago, # | +2

Nice, I'll come back here when I solve those problems.
→ Reply

Phantom2611

2 months ago, # | 0

can you provide editorials for the graph section as well
→ Reply

kartik8800

2 months ago, # ^ | 0

I'll see to it, however there are a lot of problems in the graph section. Maybe sometime in the future I'll write an editorial on some of the selected problems from that section.
→ Reply

LanceTheDragonTrainer

2 months ago, # | ← Rev. 7 +3

Distinct values queries can also be done using segment tree (with sorted vector in each node) in $\mathcal{O}(N \log^2 N)$.

Let's say your original array is A. For each index i, you store an index j such that $A[i] = A[j]$ But $i < j$. Let the array of j values be B. Build segment tree over B. For each query $[L, R]$, you just need to check how many values in each of the $\mathcal{O}(\log N)$ segments have value $> R$ via binary search.

Of course, this only works if there are no updates. Also, MO's algorithm can be adapted to solve more difficult range query problems (e.g. range query for most frequent element can be done in $\mathcal{O}(Q\sqrt{N})$).

Upd: Here is my implementation of this solution.
→ Reply

kartik8800

2 months ago, # ^ | 0

Nice approach, I am guessing this should work.
Do contribute the code if you get the time for that.
→ Reply

LanceTheDragonTrainer

2 months ago, # ^ | ← Rev. 8 +8

Sure. I will implement it when I have time.

I think I have one more approach. This time it has time complexity of $\mathcal{O}(N \log N)$.

1. Sort the queries by left endpoint.
2. For each index i, you store an index j such that $A[i] = A[j]$ But $i < j$. Let the array of j values be B.
3. Create an array of 0s (with the same size as A). Call this array C. Add 1 to $C[i]$ if i is the leftmost index of the value $A[i]$ in A. Build a segment tree/fenwick tree over C.
4. To handle a query $[L, R]$, set $C[B[i]]$ to 1 for $i < L$. Notice that we only need to do this once for each value of i so we don't need to worry about the time complexity of this operation. Then report the sum of values in $[L, R]$ using your range tree.

Once again, this approach only works if there are no updates.

This is the implementation that I wrote which got AC on CSES.
→ Reply

Priyam2k

2 months ago, # ^ | ← Rev. 2 +8

We can also solve Distinct Value Queries using Persistent Segment Tree (Online) in O(NlogN). Code — https://pastebin.com/WhkF5cCp

→ Reply

LanceTheDragonTrainer

2 months ago, # ^ | ← Rev. 8 0

Ah yes we can. I think the approach is very similar to my first segment tree solution but we sort the values in B in ascending order and set the i^{th} index to 1 in the j^{th} version of the persistent segment tree for each $B[i] = j$. Thereafter, we just need to take the difference between sum of elements in $[L, R]$ in the $(n+1)^{th}$ and R^{th} version of the segment tree for query $[L, R]$.

Even though this works, persistence is unnecessary.

→ Reply

joseacaz

2 months ago, # ^ | 0

It can also be done using a persistent segment tree in $O(NlogN + QlogN)$, here's how:

Build a persistent segment tree like this:

- Traverse the array from left to right.
- For the element with index N, update its value in the ST to 1 and save its position using a bucket (you can use a map or coordinate compression, whichever you prefer).
- For the remaining elements, if it is the first appearance of the element, update its value in the ST to a 1. Otherwise, first update the value of the position where it last appeared to 0 and then update its value in the ST to a 1.
- Save the position of this element using a bucket.

Example: For the array $[2, 3, 1, 3, 2]$, we will get the following STs:

Element 5: $[0, 0, 0, 0, 1]$

Element 4: $[0, 0, 0, 1, 1]$

Element 3: $[0, 0, 1, 1, 1]$

Element 2: $[0, 1, 1, 0, 1]$

Element 1: $[1, 1, 1, 0, 0]$

Note that since this is a persistent segment tree, you will have all different versions stored in memory.

Now, to answer a query in the form $[a, b]$, we just have to do a "sum" query from a to b on the ST of element a. This works because, by construction, only one element for every value will have a 1, and this will be the leftmost one. If we have one or more elements in the range $[a, b]$ with a certain value, the "sum" query will count only one of them. If we don't have any elements with a certain value, the "sum" query will not count them.

Total time complexity: $O(NlogN + QlogN)$, for the construction of the persistent ST and for every query.

Total memory complexity: $O(NlogN)$, because we are using a persistent segment tree.

Implementation

→ Reply


LanceTheDragonTrainer

2 months ago, # ^ | 0

Ok. This is the online version of my fenwick tree solution.
→ Reply


UnbearableREDS

2 months ago, # | 0

loved it thanks a lot for your time
→ Reply


kartik8800

2 months ago, # ^ | 0

glad to know you find this helpful.
→ Reply


zeph0yr

2 months ago, # | +3

Thank you so much **kartik8800** You did a Greatjob! I Bookmarked this page!!
→ Reply


kartik8800

2 months ago, # ^ | 0

I'm glad you liked it. thanks for the appreciation :)
→ Reply


knakul_853

2 months ago, # | 0

for `List Removals` life can be easier with this data structure
https://codeforces.com/blog/entry/10355
→ Reply


UnbearableREDS

2 months ago, # ^ | 0

have you implemented it using that? can you please share the code?
→ Reply


knakul_853

2 months ago, # ^ | 0

Spoiler
→ Reply


UnbearableREDS

2 months ago, # ^ | 0

thanks a lot can you just explain me this data structure? i tried studying from the editorial but got nothing. it would be really nice of you
→ Reply


knakul_853

2 months ago, # ^ | 0

have you went through https://en.wikipedia.org/wiki/Rope_(data_structure) let me know if I can further clarify
→ Reply


UnbearableREDS

2 months ago, # ^ | 0

yes still couldn't understand
→ Reply


UnbearableREDS

2 months ago, # ^ | 0

and also why are you not using inserting
→ Reply


UnbearableREDS

2 months ago, # | +8

Hello just a doubt that how do i build my segment tree for this question : Range Update Queries like i will be using lazy propagation for the range updates, ADDEND to increase value in the range and getValue to get the value at the kth index. Can you please help. https://ideone.com/h2Mo9u this is my segment tree template

→ Reply


emorgan5289

2 months ago, # ^ | ← Rev. 2 0

You don't need lazy prop for range sum updates and point queries. Instead of storing the array directly, store the differences between consecutive elements. Then each update can be expressed as two point updates at the ends of the interval, and each query is just a prefix sum query on the new array, which is easily solvable with a BIT or segment tree without lazy prop.

→ Reply


UnbearableREDS

2 months ago, # ^ | +8

thank you for replying yeah i have solved this question using segment tree already similar to author one, but i wanted to see if range increment(addend in case of lazzy prop can be used) to solve this question however i am unable to do it. here is the link to my template can you please guide me on solving this using lazy prop (i somehow want to use that addend function in my template) https://ideone.com/h2Mo9u

→ Reply


emorgan5289

2 months ago, # ^ | 0

Yeah, of course you can solve it with range updates and point queries on a segment tree using lazy prop. If you are learning how to do this for the first time, I suggest reading this starting from page 245. Go through the implementation carefully and test your code along the way. It's also ok to look at other people's implementations of segment trees to get an idea of how it should work.

→ Reply


kartik8800

2 months ago, # ^ | 0

This is possibly the best theory about lazy propagation that I have read till now : https://codeforces.com/blog/entry/44478?#comment-290116

Give it a read, you might find implementing lazy propagation easier.

→ Reply


Boredom

2 months ago, # | ← Rev. 3 0

You can solve `Salary Queries` offline with binary search and bit. here's my code https://pastebin.com/VG6JZaZt .

→ Reply


sillyboi

2 months ago, # ^ | 0

Can u explain your approach, like if you are using binary search on answer then how are u managing to check for this value whether it can be answer or not?

→ Reply


Boredom

2 months ago, # ^ | ← Rev. 3 0

We can add all the salaries that appear in the initial salaries array & in the queries to a vector ve. Updates: Let x be the salary of an employee we will get the index k of x in ve and add one to k in the BIT, we'll do the same thing to subtract the old salary of this employee from the BIT.Queries: we will get the indices of l and r in ve using bs and answer the queries from

bit.

time complexity : O(nlogn+qlogn)
→ Reply


emorgan5289

2 months ago, # | +3

Salary Queries can be solved for arbitrarily large values in the array using coord compression to get all values in the range $[1..N]$, then just use a BIT, segment tree, or whatever your favorite data structure is to solve it. The implementation is very messy, so I wouldn't recommend coding it, but it ends up taking $O(n \log n + q \log n)$.
→ Reply


kartik8800

2 months ago, # ^ | 0

Seems reasonable but can you explain what happens when the salary of an employee gets updated?

Let's say I did the coordinate compression on the salaries [2,6,8] and got [1,2,3]. Now the query says change 6 to 3, and 6 is mapped to 2, what do you change the compressed_number(2) to?

Little confused on how you will work with updates.
→ Reply


emorgan5289

2 months ago, # ^ | ← Rev. 2 +6

You have to take into account future updates when running the coord compression. For instance, append all future values to the array, compress it, and then remove them again. You would also do the same thing for both endpoints of all queries.

So I guess the complexity is actually $O((n+q)\log(n+q))$, this is basically the same thing though given the constraints.
→ Reply


kartik8800

2 months ago, # ^ | 0

Ah, got it!

Makes complete sense, thanks for sharing.
→ Reply


luismo

5 weeks ago, # ^ | 0

That was my approach and it led me to TLE, I'm using a set and a map to compress, and a FenwickTree to perform queries later on. I haven't tried compressing with vectors + binary search, but I don't think it would make a difference, any help will be appreciated
→ Reply


luismo

5 weeks ago, # ^ | 0

Well apparently compressing with vectors + binary search is allowed to pass meanwhile set + maps isn't, since my code is AC now after that modification
→ Reply


emorgan5289

5 weeks ago, # ^ | 0

I got AC while using a map for compression. Here's what I used:
Code
→ Reply

luismo

5 weeks ago, # ^ | 0

Thanks

→ Reply

lil_ninja

2 months ago, # | 0

Anyone has any idea where to find editorial of String Algorithm section of CSES problemset?

→ Reply

T--o_o--T

5 weeks ago, # | 0

please anyone give me HOTEL QUERIES problem solution.

→ Reply

Jon.Snow

5 weeks ago, # | +5

xor queries can also be solved using
$xor(l, r) = xor(xor(1, r), xor(1, l - 1))$
right?

→ Reply

kartik8800

5 weeks ago, # ^ | +3

sure it will work perfectly and in O(n) time.
here is the AC code : https://ideone.com/MuIx1L
thanks for mentioning.

→ Reply

sagsango

5 weeks ago, # | 0

Range Sum Queries I
Range Minimum Queries I
Range Sum Queries II
Range Minimum Queries II
Range Xor Queries
Range Update Queries
Forest Queries
Hotel Queries
List Removals
Salary Queries
Subarray Sum Queries
Distinct Values Queries
Forest Queries II
Range Updates and Sums
Polynomial Queries
Range Queries and Copies

→ Reply

It_Wasnt_Me

5 weeks ago, # | ← Rev. 3 +5

Just mention that "Distinct Values Queries" problem have segment tree solution

code

→ Reply

Priyansh31dec

4 days ago, # | 0

I tried solving the problem Salary Queries by the way that you have suggested. I used Fenwick Tree instead of a Segment Tree. I am getting TLE in it. However, when I tried running the inputs, the answers are coming out to be correct. Any suggestions on optimizing it? Here is the link to my code: https://ideone.com/dtWevB

→ Reply

4 days ago, # ^ | +3

kartik8800

on first look, I would say it is possible that your helper function is causing the TLE.

Accessing an element of a map is O(logN) operation. so inside your helper you will access it 100 times -> 100*log(N), in my implementation I have eliminated this logN factor. and my helper takes 100 + logN.

So might be the reason for TLE, try optimizing and let me know if it passes.

→ Reply

Priyansh31dec

4 days ago, # ^ | 0

So, do you mean to say that instead of creating a map of <int,int> I should create a map of <int,vector> so that I can first access the map element in logN and then traverse the vector in 100 operations?

→ Reply

kartik8800

4 days ago, # ^ | +3

Read the calc() method of my Implementation.
I do one map access to get iterator corresponding to value lo. From there onwards I increment the iterator till the key being pointed by iterator is less than hi.

Iterator increment is O(1) operation.

→ Reply

Priyansh31dec

4 days ago, # ^ | +8

Doing this worked out for me. Thanks a lot for your help. This question really taught me a lot. Thanks again.

→ Reply

kartik8800

4 days ago, # ^ | +3

Glad to know, You're welcome :)

→ Reply

Codeforces (c) Copyright 2010-2020 Mike Mirzayanov
The only programming contests Web 2.0 platform
Server time: Jul/14/2020 15:08:05 UTC+5.5 (i3).
Desktop version, switch to mobile version.
Privacy Policy

Supported by

ITMO UNIVERSITY