Eureka!!

Thursday, July 21, 2011

Segment Tree

Segment trees (for beginners)

Lets start with a simple problem.

Given an array of numbers. There are 2 types of operations to do.
1. update any element in the array
2. find the maximum/minimum in the given range (i,j)

```
Simple solution.
-- updating is no problem. a[i]=x; :D
-- finding maximum/minimum can be done in one for loop of (j-i+1)
iterations.
```

Can we do any better?
The answer is yes.
In fact we can do both the operation in `O(log n)` where n is the size of the array.

That's when segment tree comes handy.

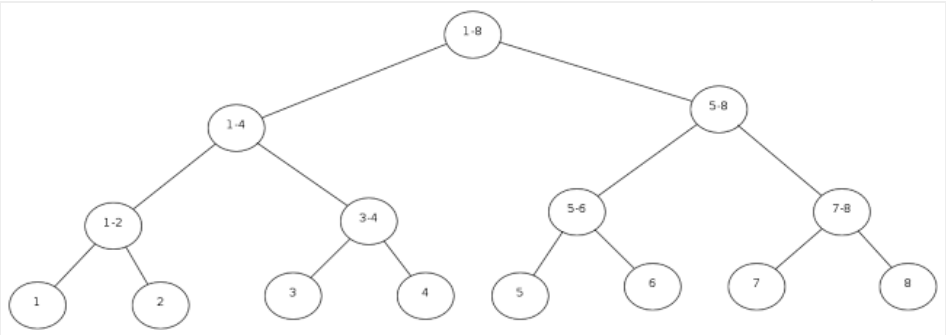
A **Segment Tree** is a **binary tree**.
For the sake of convinience, lets take an array of size `N = 2^n`.

Suppose we have some work to do. If one person does it all, a lot of effort/time is needed.
So to reduce the time as well as effort we can divide the responsibilities among many people.

We can divide the responsibilities in **Segment Trees** as well.
Lets distribute the responsibilities to different nodes as follows.
-> each node carries the maximum and/or minimum value of all its childrens.

For example:
Suppose we have an array of size 8.

Let us construct a complete binary tree (segment tree) as below.



In a segment tree the list of leaf nodes represents the actual array where the main data resides.
Here there are 8 leaf nodes which correspond to the elements of array we wish to update or find max/min in the range.

Each node stores the required information (here maximum/minimum) assigned to them.
for eg: The root node stores maximum/minimum of all the nodes from node 1 to node 8.

About Me

ashish pant
Follow 643

I am Ashish Pant.
View my complete profile

Table Of Content

Followers

Assume such tree exists. Lets walk through how we process the following queries.

1. Find maximum in range (1,8)

Return the answer stored in the root of the tree.

2. Find maximum in range (3,6)

```
check root(1-8) ---(our range is a subset of this range) so branch in.
check (1-4) hmm..contains partially
check (1-2) out of range...dont check its children.
check (3-4) hmm..completely inside the range...take the value..dont go
to children(since it lies completely inside the range (3,6) ).
check (5-8) hmm...contains our answer partially
check (5-6) completely inside our range. so take its info. dont branch
in.
check (7-8) completely outside our range. dont branch in.
```

Now finally compare all the values we took in the process.
Remember we took the values of range(3-4) and (5-6)
Just compare them and we have the maximum.

Here we visited none of the leaf nodes. We visited some internal nodes only.
In practice if the segment tree is very large then we skip too many nodes.
So effectively the number of nodes examined will be of order of O(height) the tree.
which is log N.

**********Happy Coding**********

Let me give you how to build the tree for finding maximum in a range.
A binary tree can be represented by an array. For convenience I have named the root as 1. The 2 children of a node n are 2*n and 2*n + 1.

If you have an array 'arr' of size 1000, then call build function as below.
build (1, 0, 999);

```
int tree[3*MAX]; //a segment tree is nearly twice as large as the
array.
int arr[MAX];

void build(int n, int b, int e)
{
 if (b>e) return;
 else if (b==e)
 {
  tree[n] = arr[b];
  return;
 }

 build ( n*2 , b , (b+e)/2 ); //go to children...child nodes of node n
are 2n and 2n+1.

build (n*2+1, (b+e)/2 + 1 , e );

 //now both child nodes 2n and 2n+1 are built (ie they have done their
responsibility of storing the correct information)
 tree[n] = max( tree[n*2] , tree[n*2 + 1] );

}



/*
* update in the tree at index idx with value val.
* remember here n is the node number of the tree and not index of
array...value of root node 1 and its children are 2 and 3
* idx is the mapping of the leaf nodes to array. when b==e we reached
leaf node
*/
void update(int n, int b, int e, int idx, int val)
{
```

```
 if (b>e || b>idx || e<idx ) return;
 if (b==e) //at leaf node
 {
  tree[n] = val;
  return;
 }

 update( n*2 , b , (b+e)/2 , idx, val );
 update( n*2 + 1 , (b+e)/2 + 1 , e , idx, val );

 //now some change might have been made in either of the child nodes.

 tree[n] = max( tree[n*2] , tree[n*2 + 1] );

}


//query function is homework.

int query(int n, int b, int e, int i, int j) //return the maximum in
the range (i,j)
{
...
...
}

/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/
```

The problem of finding maximum in the range requires a little more effort.
Reason:
-> lazy propagation updates only at top level nodes.
-> each node can **either** store increments done to the range **or** only the maximum.
having only one of these is not sufficient.

Thus we need to store both the information.
store 2 variables. one is maximum of the child nodes and another is total increment done to the node.

the actual maximum number at each node is
maximum number stored in a node + total increments to the node and all its parents.

Below is full source code of the program.

```
#include < cstdio >
#include < iostream >
using namespace std;

#define INF 1e9

struct Node {
 int offt;
 int cmax;
} tree[5000];

void update(int n, int b, int e, int i, int j, int val)
{
 if (b>e || i>j || b>j || e

 if (b>=i && e<=j)
 {
  tree[n].offt += val;
  tree[n].cmax += val;
  return;
 }

 update(n*2 , b , (b+e)/2 , i , j , val);
 update(n*2+1 , (b+e)/2 + 1 , e , i , j , val);

 tree[n].cmax = max ( tree[n*2].cmax , tree[n*2+1].cmax ) +
tree[n].offt;
```

```
}

int query(int n, int b, int e, int i, int j, int offt)
{
 if (b>e || i>j || b>j || e

 if (b>=i && e<=j)
  return tree[n].cmax + offt; //the increment of current node is
already added to cmax here (see update function)

 offt += tree[n].offt;
 return max ( query(n*2 , b , (b+e)/2 , i , j, offt) ,
  query(n*2+1 , (b+e)/2 + 1 , e , i , j, offt) );
}


int main()
{
 int tc,x,a,b,v;
 cin >> tc;
 while(tc--)
 {
  cin >> x >> a >> b;
  if ( x == 0 )
  {
   cin >> v;
   update(1,0,999,a,b,v);
  }
  else
   cout << query(1,0,999,a,b,0) << endl;
 }
}
```

****

Posted by ashish pant at 1:47 AM

+6 Recommend this on Google

35 comments:

geek July 21, 2011 at 5:00 AM

plz xplain a bit more how update is implemented

Reply

geek July 21, 2011 at 6:49 AM

This comment has been removed by a blog administrator.

Reply

P!=NP July 21, 2011 at 9:02 AM

small typing error. now corrected.

Reply

geek July 21, 2011 at 10:20 AM

what is the use of idx here in update function??

Reply

P!=NP July 21, 2011 at 1:11 PM

idx maps to the index in the array.
basically (b,e) is a range represented by node n
so when b=e we reach the leaf node corresponding to index b (which at that time also equals idx) of the array.

Reply

shadow July 22, 2011 at 11:40 PM

in 2nd prob. in update section ....shouldn't the line "tree[n]+=val" be "tree[n]+=(e+1-b)*val"...

Reply

Amol Sharma July 29, 2011 at 10:08 PM

sir,in the second illustration you gave just after the tree diagram....i think you have mistyped the query

it should have been
find maximum in range (3,6)

Reply

Replies

ashish pant January 22, 2012 at 5:34 AM

thanks amol

Reply

geek July 31, 2011 at 1:57 AM

```
int query(int n,int b,int e,int i,int j)
{
if(i>e || j=i && e<=j) return tree[n];

int p1=query(2*n,b,(b+e)/2,i,j);
int p2=query(2*n+1,(b+e)/2+1,e,i,j);

if(p1==-1) return p2;
if(p2==-1) return p1;
if(p1>p2) return p1;
else return p2;

}
```

Reply

geek July 31, 2011 at 1:59 AM

sir how to impement this with lazy propogation.....
plz xpain a bit more.... don post the code for now

Reply

mad_coder January 24, 2012 at 1:26 AM

Sir, i have implemented the first query function as in the code given below but with extra space complexity. Is there a way to reduce the extra space complexity ?

```
int arr[MAX],tree[2*MAX],max_arr[2*max];
int query(int n,int b,int e,int i,int j)
{
if(b>e||b>j||e=i&&e<=j)
return tree[n];
max_arr[2*n]=query(2*n,b,(b+e)/2,i,j);
max_arr[2*n+1]=query(2*n+1,(b+e)/2+1,e,i,j);
return max(max_arr[2*n],max_arr[2*n+1]);
}
```

Reply

Replies

ashish pant January 24, 2012 at 5:13 AM

it think there is some mistake...what is tree[n]? why is it not updated? may be i need to look at the update function.

ashish pant January 24, 2012 at 5:13 AM

And dont call me sir. Just post your question. :)

Reply

Nitin February 12, 2012 at 9:33 AM

Nice tutorial man. Really helpful :)

Reply

CSL August 2, 2012 at 11:12 AM

where is the full code sir

Reply

Chan Nguyen August 4, 2012 at 3:44 PM

Hi Ashish Pant, the update function doesn't seem to work properly.
I wonder if you could explain what is `b` and `e`? Is it the full range of the data array? If I have an array data[10], then will update is called as update(1, 0, 10, left, right, some-value)? Thanks.

Reply

▼ Replies

ashish pant September 3, 2012 at 10:40 PM

call update (1, 0, 9, left, right, some-value)
Here b and e stand for begin and end inclusive.

Reply

swat August 11, 2012 at 7:42 AM

thanks dude. nice code.

Reply

Atul anand September 23, 2012 at 10:42 AM

Hi could you please explain the following :-
struct Node {
int offt;
int cmax;
} tree[5000];

what is offt and cmax holding ?
when i initalize the tree , then what value should be stored at offt and cmax?
in the query function query(1,0,999,a,b,0);// last parameter is offt , what is the use of this parameter.

Thanks in advance

Reply

▼ Replies

ashish pant September 24, 2012 at 7:11 PM

offt is the total increments done to the subtree represented by the node.
Thus if one needs to find how many times certain node is incremented we need to sum the offts of all its parents.
cmax is the updated maximum value after updates of the subtree represented by the node.
cmax is updated on every updates to the node.

Reply

karim November 12, 2012 at 12:01 PM

hi ashish i really like your tutorial specially the part with lazy propagation but i noticed that the full source code for the program is not entirely correct (not the missing parts) but in the update function

1 - update(n*2 , b , (b+e)/2 , i , j , val);
2 - update(n*2+1 , (b+e)/2 + 1 , e , i , j , val);
3 - tree[n].cmax = max (tree[n*2].cmax , tree[n*2+1].cmax);

the 3rd line i think we should add the tree[n].offt to the maximum as this maximum is not propagated so it would give incorrect results if we query this exact interval again

Reply

Replies

ashish pant November 12, 2012 at 2:54 PM

Hi Karim,
Thank you for your interest in my blog.
I think my solution is correct and consider this an incompetence on my part not to be able to explain the logic properly. If you find any test case where my solution fails, please post it here for I don't want to convey any wrong solution to the public.

As for the explanation of your question, the idea is to add the increments to only those nodes whose all children completely lie inside the range. Thus if you take a path from any one leaf node (any one array element from the range where the increment is to be done) to the root, you will find that the increment is done to only one node in the path. This helps the prevent addition of a certain increment more than once.

karim November 16, 2012 at 9:08 AM

ok here is a query

update [1-10]
update [1-5]
query [1-10]

here update denotes increment by 1

so the second query would be broken into two subqueries
update [1-5] and update [6-10]
and [1-10] = max([1-5], [6-10])

query [1-10] will return the new value without the total offt at [1-10] so i think the update should be as follows

tree[n].cmax = max(tree[n*2].cmax,tree[n*2+1].cmax) + tree[n].offt;

ashish pant November 16, 2012 at 10:01 AM

oh right. Thank you very much karim. I am grateful that you pointed out the mistake.

Reply

Ravin November 13, 2012 at 3:21 PM

Hi, could you please provide query function for range modification and range sum, because I don't get it.

For example we got array with size 10 and two queries:
1) add to whole array 3
2) get sum from element 4 to 5 inclusive

According to your code we'll call update(1, 0, 9, 0, 9, 3) and it will update first node and break; so question is how do we implement sum function here using nlog time? thanks :)

Reply

Replies

ashish pant November 16, 2012 at 10:08 AM

I also pointed out that we need a new member variable in every node that denotes how much increments/decrements have been done to all the children of that node.
The query function should be very similar to the solution I have provided at the bottom of the page

Reply

Panagiotis Kostopanagiotis January 2, 2013 at 7:11 AM

nice one.. pretty straight forward article with nice explanations.
Segment tree is a really important data structure for programming competitions , and articles like this are really helpful!

Thanks :)

Reply

Viki Sharma February 7, 2013 at 8:55 PM

Wonderful blog! Really useful :)
I had a doubt with a particular question involving segment trees:
http://www.spoj.com/problems/IOPC1207/
What I can't understand is why we can use 3 independent segment trees. If we update the x range say x1 to x2, won't the corresponding segment trees on y and z axis also get affected?

Reply

Mayank May 9, 2013 at 3:56 AM

really helpful blog. Loved how simple the explanation was. I had a doubt though. If i need to do a query for a range and find both the min and max in that range what should be done? I made two trees with one build function: one containing the maximums and the other containing minimums. But i cant seem to figure out how to do both the jobs with one query function i am having to call two separate functions one for maximum and one for minimum.

Thanks

Reply

Replies

ashish pant May 9, 2013 at 8:23 AM

For both min and max all you need is to add one more variable to each tree node to store the minimum. And add few lines to update and query function. It should be quite similar to updating the maximum.

Reply

Mayank May 11, 2013 at 8:52 AM

ya got that...silly on my part :D
thanx a lot!

Reply

The Author December 8, 2013 at 9:11 AM

Dude, you're a life saver. Been looking all over on how to query the goddamn tree and it finally makes sense!.
Thanks a ton!

Reply

Replies

Adrian Carballo December 21, 2013 at 1:12 PM

Here is another nice tutorial, implemented in python http://sportcoder.com/segment-trees/

Reply

Rahul Kumar December 18, 2014 at 1:23 AM

we have to take array size of 2^n for make segment tree of n element array,then how to make segment tree of 30 or greater elements,because 2^30 = 1073741824
then how to take array of this larger size for make segment tree,how to implement ?

Reply

Nidhi Makhijani February 15, 2015 at 2:44 AM

great post!

Reply

Enter your comment...

Comment as: Google Account

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)