My Experience

Monday, December 24, 2012

Segment Trees and lazy propagation

In this topic i will explain a very interesting data structure that can be used to solve a specific set of problems. I will start by explaining its definition and the proceeding with an example problem to solve with it.

Table of contents:

What is segment trees?

Segment Trees is a Tree data structure for storing intervals, or segments, It allows querying which of the stored segments contain a given point. It is, in principle, a static structure; that is, its content cannot be modified once the structure is built. It only uses O(N lg(N)) storage.

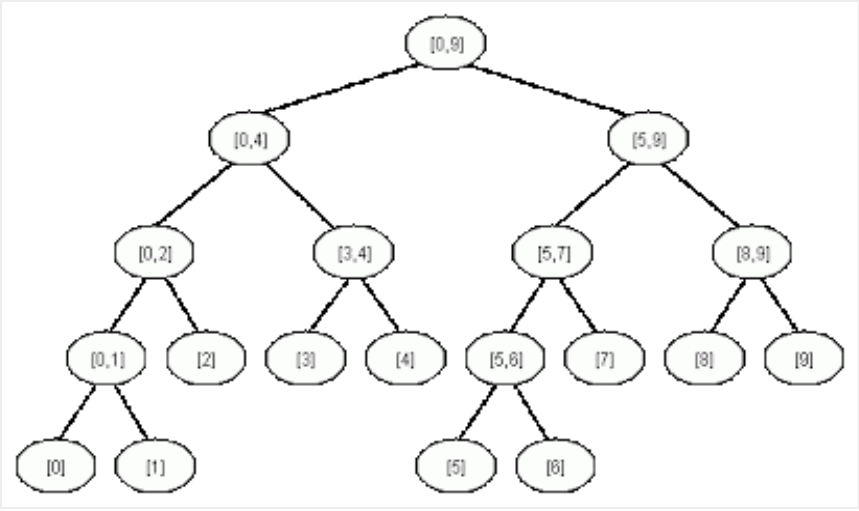
A segment trees has only three operations: build_tree, update_tree, query_tree.

Building tree: To init the tree segments or intervals values
Update tree: To update value of an interval or segment
Query tree: To retrieve the value of an interval or segment

Example Segment Tree:

- The first node will hold the information for the interval **[i, j]**
- If i<j the left and right son will hold the information for the intervals **[i, (i+j)/2]** and **[(i+j)/2+1, j]**

Notice that the height of a segment tree for an interval with **N** elements is **[logN] + 1**. Here is how a segment tree for the interval **[0, 9]** would look like:



Order of growth of segment trees operations

- **build_tree:** O(N lg(N))
- **update_tree:** O(lg(N + k))
- **query_tree:** O(lg(N + k))

K = Number of retrieved intervals or segments

Show me your code

```
/**
 * In this code we have a very large array called arr, and very large set of
 * Operation #1: Increment the elements within range [i, j] with value val
 * Operation #2: Get max element within range [i, j]
 * Build tree: build_tree(1, 0, N-1)
 * Update tree: update_tree(1, 0, N-1, i, j, value)
 * Query tree: query_tree(1, 0, N-1, i, j)
 */

```


► 2013 (4)

▼ 2012 (17)

▼ December (4)

Segment Trees and lazy propagation

Lucas' Theorem to solve binomial coefficients

Hacking linux keyboard

Apache JMeter along with jsf pages

► October (3)

► September (1)

► July (1)

► April (5)

► January (3)

► 2011 (10)

About Me

Hussein El-Sayed

Cairo, Egypt

Software engineer at Vodafone international services, topcoder handle is husseincoder

View my complete profile

```cpp
#include<iostream>
#include<algorithm>
using namespace std;

#include<string.h>
#include<math.h>

#define N 20
#define MAX (1+(1<<6)) // Why? :D
#define inf 0x7fffffff

int arr[N];
int tree[MAX];

/**
 * Build and init tree
 */
void build_tree(int node, int a, int b) {
    if(a > b) return; // Out of range

        if(a == b) { // Leaf node
                tree[node] = arr[a]; // Init value
                return;
        }

        build_tree(node*2, a, (a+b)/2); // Init left child
        build_tree(node*2+1, 1+(a+b)/2, b); // Init right child

        tree[node] = max(tree[node*2], tree[node*2+1]); // Init root value
}

/**
 * Increment elements within range [i, j] with value value
 */
void update_tree(int node, int a, int b, int i, int j, int value) {

        if(a > b || a > j || b < i) // Current segment is not within range [i
                return;

        if(a == b) { // Leaf node
                tree[node] += value;
                return;
        }

        update_tree(node*2, a, (a+b)/2, i, j, value); // Updating left child
```

```
55          update_tree(1+node*2, 1+(a+b)/2, b, i, j, value); // Updating right c
56
57          tree[node] = max(tree[node*2], tree[node*2+1]); // Updating root with
58  }
59
60  /**
61   * Query tree to get max element value within range [i, j]
62   */
63  int query_tree(int node, int a, int b, int i, int j) {
64
65          if(a > b || a > j || b < i) return -inf; // Out of range
66
67          if(a >= i && b <= j) // Current segment is totally within range [i, j
68                  return tree[node];
69
70          int q1 = query_tree(node*2, a, (a+b)/2, i, j); // Query left child
71          int q2 = query_tree(1+node*2, 1+(a+b)/2, b, i, j); // Query right chi
72
73          int res = max(q1, q2); // Return final result
74
75          return res;
76  }
77
78  int main() {
79          for(int i = 0; i < N; i++) arr[i] = 1;
80
81          build_tree(1, 0, N-1);
82
83          update_tree(1, 0, N-1, 0, 6, 5); // Increment range [0, 6] by 5
84          update_tree(1, 0, N-1, 7, 10, 12); // Incremenet range [7, 10] by 12
85          update_tree(1, 0, N-1, 10, N-1, 100); // Increment range [10, N-1] by
86
87          cout << query_tree(1, 0, N-1, 0, N-1) << endl; // Get max element in
88  }
```

segment_tree.cpp hosted with by **GitHub** **view raw**

Lazy Propagation

Sometimes a segment tree operation wouldn't survive if the problem constraints is too large, here it come lazy propagation along with the segment tree.

In the current version when we update a range, we branch its childs even if the segment is covered

within range. In the lazy version we only mark its child that it needs to be updated and update it when needed.

Note: Read my solution for problem Can you answer these queries I in this article.

```
/**
 * In this code we have a very large array called arr, and very large set of
 * Operation #1: Increment the elements within range [i, j] with value val
 * Operation #2: Get max element within range [i, j]
 * Build tree: build_tree(1, 0, N-1)
 * Update tree: update_tree(1, 0, N-1, i, j, value)
 * Query tree: query_tree(1, 0, N-1, i, j)
 */

#include<iostream>
#include<algorithm>
using namespace std;

#include<string.h>
#include<math.h>

#define N 20
#define MAX (1+(1<<6)) // Why? :D
#define inf 0x7fffffff

int arr[N];
int tree[MAX];
int lazy[MAX];

/**
 * Build and init tree
 */
void build_tree(int node, int a, int b) {
        if(a > b) return; // Out of range

        if(a == b) { // Leaf node
                tree[node] = arr[a]; // Init value
                return;
        }

        build_tree(node*2, a, (a+b)/2); // Init left child
        build_tree(node*2+1, 1+(a+b)/2, b); // Init right child

        tree[node] = max(tree[node*2], tree[node*2+1]); // Init root value
```

```
}

/**
 * Increment elements within range [i, j] with value value
 */
void update_tree(int node, int a, int b, int i, int j, int value) {

        if(lazy[node] != 0) { // This node needs to be updated
                tree[node] += lazy[node]; // Update it

                if(a != b) {
                        lazy[node*2] += lazy[node]; // Mark child as lazy
                        lazy[node*2+1] += lazy[node]; // Mark child as lazy
                }

                lazy[node] = 0; // Reset it
        }

        if(a > b || a > j || b < i) // Current segment is not within range
                return;

        if(a >= i && b <= j) { // Segment is fully within range
                tree[node] += value;

                if(a != b) { // Not leaf node
                        lazy[node*2] += value;
                        lazy[node*2+1] += value;
                }

                return;
        }

        update_tree(node*2, a, (a+b)/2, i, j, value); // Updating left chil
        update_tree(1+node*2, 1+(a+b)/2, b, i, j, value); // Updating right

        tree[node] = max(tree[node*2], tree[node*2+1]); // Updating root wit
}

/**
 * Query tree to get max element value within range [i, j]
 */
int query_tree(int node, int a, int b, int i, int j) {

        if(a > b || a > j || b < i) return -inf; // Out of range

```

```
85          if(lazy[node] != 0) { // This node needs to be updated
86                  tree[node] += lazy[node]; // Update it
87
88                  if(a != b) {
89                          lazy[node*2] += lazy[node]; // Mark child as lazy
90                          lazy[node*2+1] += lazy[node]; // Mark child as lazy
91                  }
92
93                  lazy[node] = 0; // Reset it
94          }
95
96          if(a >= i && b <= j) // Current segment is totally within range [i,
97                  return tree[node];
98
99          int q1 = query_tree(node*2, a, (a+b)/2, i, j); // Query left child
100         int q2 = query_tree(1+node*2, 1+(a+b)/2, b, i, j); // Query right ch
101
102         int res = max(q1, q2); // Return final result
103
104         return res;
105 }
106
107 int main() {
108         for(int i = 0; i < N; i++) arr[i] = 1;
109
110         build_tree(1, 0, N-1);
111
112         memset(lazy, 0, sizeof lazy);
113
114         update_tree(1, 0, N-1, 0, 6, 5); // Increment range [0, 6] by 5
115         update_tree(1, 0, N-1, 7, 10, 12); // Incremenet range [7, 10] by 12
116         update_tree(1, 0, N-1, 10, N-1, 100); // Increment range [10, N-1] b
117
118         cout << query_tree(1, 0, N-1, 0, N-1) << endl; // Get max element in
119 }
```

lazy_segment_tree.cpp hosted with by **GitHub** **view raw**

Sample Problems to try

- Quadrant Queries
- D-Query
- Can You answer these queries I

References

- Wiki
- Topcoder tutorials

Posted by Hussein El-Sayed at 4:58 AM

+4 Recommend this on Google

26 comments:

donnghi December 24, 2012 at 12:25 PM

In lazy's version, i thinks it's better if you replace update tree[2*node] and tree[2*node+1] in 49th, 50th, 80th and 81th line by lazy[2*node] and lazy[2*node+1].

Its reason is your query is not really come down to higher level, so lazy[] should be updated

Reply

Hussein El-Sayed December 24, 2012 at 10:44 PM

I can't understand you :)

Reply

donnghi December 25, 2012 at 3:03 AM

So, in line 80th:
tree[node*2] += lazy[node]; // Mark child as lazy
tree[node*2+1] += lazy[node]; // Mark child as lazy

=> replaced by:
lazy[node*2] += lazy[node];
lazy[node*2+1] += lazy[node];

It's correct ?

Reply

Hussein El-Sayed December 25, 2012 at 3:29 AM

Yes you are totally right :).. thanks for correcting me ;)..

Reply

Hussein El-Sayed December 25, 2012 at 3:30 AM

The same at line 49 and 50, updated check it now and tell me :)

Reply

Sandipan Manna December 31, 2012 at 12:13 PM

update_tree(1, 0, N-1, 0, 6, 5); // Increment range [0, 6] by 5
update_tree(1, 0, N-1, 7, 10, 12); // Incremenet range [7, 10] by 12
update_tree(1, 0, N-1, 10, N-1, 100); // Increment range [10, N-1] by 100

So, The maximum element in the whole array should be 101
and your final array becomes
[6 6 6 6 6 6 6 13 13 13 13 101 101 101 101 101 101 101 101]

but your program gives output as 117 !!!

Reply

Hussein El-Sayed January 1, 2013 at 3:06 AM

No it should be 113, however the size of the array needs to be (1+(1<<6)) as it should be 2^(1+lg N)..

Also there was some checks needed to be added in the lazy version.. please check it and get back to me.

Reply

Sandipan Manna January 11, 2013 at 11:19 PM

Yes your segment tree size should be
int x = (int)(ceil(log2(N)))+1;
size = (1 << x);
This one!

Reply

InfiniteComplexity February 16, 2013 at 3:27 AM

Wow! Thanks for this post, it was very helpful! However, I'm trying to implement another update_tree_val function that sets the values from a range to one value. e.g. update_tree_val(3,7,4) would set the range [3,7] to the value of 4. How can I do this using lazy propagation on your tree?

Thanks

Reply

Replies

Hussein El-Sayed February 16, 2013 at 11:33 PM

It would be the same but without incrementing..

Aditya Choudhary November 9, 2014 at 8:58 AM

This means in line 48 ,I have to modify it to tree[node] = lazy[node];
and in line 62, tree[node] = value; ?

Reply

PRASHANTHSOUNDAR February 19, 2013 at 9:37 AM

hi, as a beginner to seg tree and lazy propagation i found this post very useful . i tried solving http://www.spoj.com/problems/HORRIBLE/
using slightly modified but same method as this but i am getting WA . can u help?..
http://ideone.com/3rBgSC

Reply

Tilak Raj Singh July 26, 2013 at 6:08 PM

in line 86 it should be
tree[node] += (b-a+1)*lazy[node];

Reply

Replies

ravi November 2, 2014 at 8:22 PM

I think you forgot that query returns maximum from given range
not the sum of elements of given range

Reply

rl September 15, 2013 at 10:15 AM

In order to do the following:
1.) Add x to A[i], A[i+1],...,A[j]
2.) Output the sum of A[i],A[i+1],...A[j]
I simply replaced max() with sum() however i am not getting the correct answer.

Reply

Replies

ravi November 2, 2014 at 8:25 PM

also replace line 48 and 86 by tree[node] += (b - a + 1) * lazy[node];
and line 62 by tree[node] += (b - a + 1) * value;

Reply

ইমতিয়াজ November 13, 2013 at 11:37 AM

This comment has been removed by the author.

Reply

Gautam Singh May 24, 2014 at 12:55 AM

in query_tree() method we could have updated the lazy value to the tree before we look for the out of range condition.... it would result in better performance...

am i right about it....please correct me if I am wrong!!

Reply

VIPUL JAIN August 1, 2014 at 7:16 AM

Can you please elaborate how to implement DQUERY?

Reply

Ashish Tilokani September 12, 2014 at 1:36 AM

http://discuss.codechef.com/questions/50866/segment-trees-doubt

Reply

Ashu Pachauri September 25, 2014 at 10:11 PM

Very helpful

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

Hussein El-Sayed December 18, 2014 at 1:27 AM

It should be 2^(1+lg N) not 2^N

Reply

ম্যাট্রিক্স.কোড January 6, 2015 at 2:36 PM

This comment has been removed by the author.

Reply

ম্যাট্রিক্স.কোড January 6, 2015 at 2:43 PM

Very good and nice blog..
I am following your code structure in my SG tree implementation :)

Reply

Jayaram Prabhu Durairaj January 7, 2015 at 7:49 PM

This comment has been removed by the author.

Reply

Enter your comment...

Comment as: Select profile...

Publish **Preview**

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Simple template. Powered by Blogger.