Segment Tree | Set 1 (Sum of given range)

Let us consider the following problem to understand Segment Trees.

We have an array arr[0 . . . n-1]. We should be able to
1 Find the sum of elements from index l to r where 0 <= l <= r <= n-1
of a specified element of the array arr[i] = x where 0 <= i <= n-1.

A
range. To update a value, simply do arr[i] = x.
second operation takes O(1) time.

Another solution is to create another array and store sum from start to i at the ith index in this array. Sum of a given range can now be calculated in O(1) time, but

update operation takes O(n) time now. This works well if the number of query operations are large and very few updates.

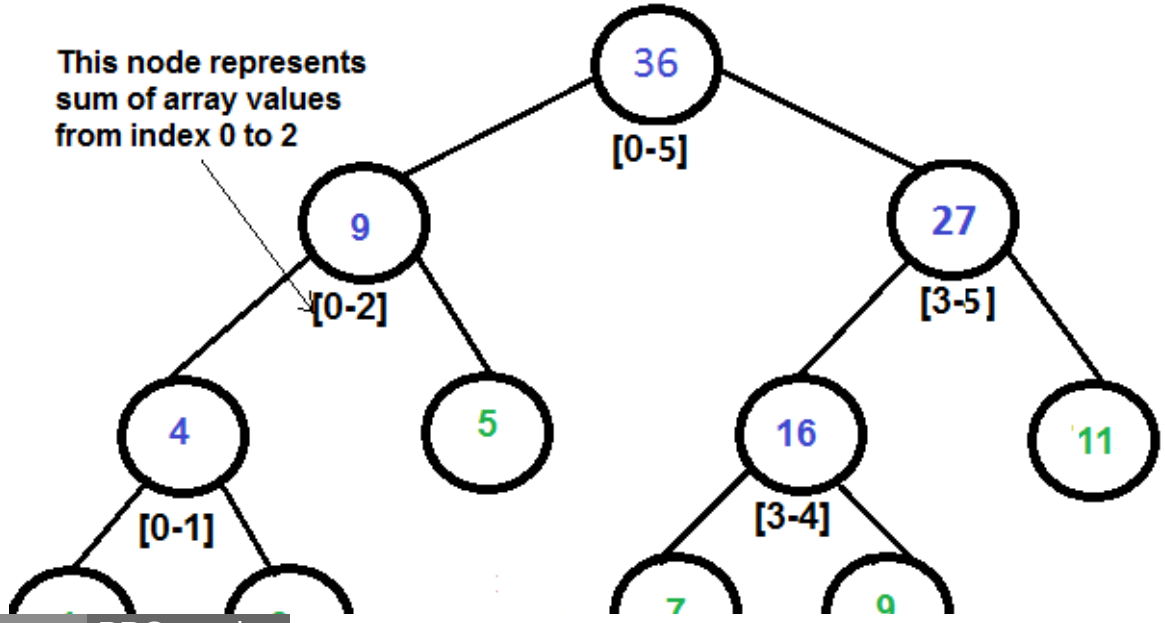
What if the number of query and updates are equal? **Can we perform both the operations in O(log n) time once given the array?** We can use a Segment Tree to do both operations in O(Logn) time.

Representation of Segment trees

1. Leaf Nodes are the elements of the input array.

2. Each internal node represents some merging of the leaf nodes. The merging may be different for different problems. For this problem, merging is sum of leaves under a node.

An array representation of tree is used to represent Segment Trees. For each node at index i, the left child is at index 2*i+1, right child at 2*i+2 and the parent is at $\lfloor (i-1)/2 \rfloor$.



Popular Posts

- Top 10 Algorithms and Data Structures for Competitive

Segment Tree for input array {1, 3, 5, 7, 9,11}

Construction of Segment Tree from given array

We start with a segment arr[0 . . . n-1]. and every time we divide the current segment into two halves(if it has not yet become a segment of length 1), and then call the same procedure on both halves, and for each such segment we store the sum in corresponding node.

All levels of the constructed segment tree will be completely filled except the last level. Also, the tree will be a

two halves at every level. Since the constructed tree is always full binary tree with n leaves, there will be n-1 internal nodes. So total number of nodes will be 2*n – 1.

Height of the segment tree will be $\lceil \log_2 n \rceil$. Since the tree is represented using array and relation between parent and child indexes must be maintained, size of memory allocated for segment tree will be $2 * 2^{\lceil \log_2 n \rceil} - 1$.

Query for Sum of given range

Once the tree is constructed, how to get the sum using the constructed segment tree. Following is algorithm to get the sum of elements.

```
int getSum(node, l, r)
{
   if range of node is within l and r
        return value in node
```

Programming
- Top 10 algorithms in Interview Questions
- How to begin with Competitive Programming?
- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Heavy Light Decomposition
- Sorted Linked List to Balanced BST
- Comb Sort
- Aho-Corasick Algorithm for Pattern Searching

```
    else if range of node is completely outside l and r
         return 0
    else
     return getSum(node's left child, l, r) +
            getSum(node's right child, l, r)
}
```

Update a value

Like tree construction and query operations, update can also be done recursively. We are given an index which needs to updated. Let us
start from root of the segment tree, and add *diff* to all nodes which have given index in their range. If a node doesn't have given index in its range, we don't make any changes to that node.

Implementation:

Following is implementation of segment tree. The program implements construction of segment tree for any given array. It also implements query and update operations.

C Java

```
// C program to show segment tree operations like construct
ion, query
// and update
#include <stdio.h>
#include <math.h>

// A utility function to get the middle index from corner i
```

```
ndexes.
int getMid(int s, int e) {  return s + (e -s)/2;  }

/*  A recursive function to get the sum of values in given
range
    of the array. The following are parameters for this fun
ction.

    st    --> Pointer to segment tree
    si    --> Index of current node in the segment tree. In
itially
              0 is passed as root is always at index 0
    ss & se  --> Starting and ending indexes of the segment
 represented
                 by current node, i.e., st[si]
    qs & qe  --> Starting and ending indexes of query range
 */
int getSumUtil(int *st, int ss, int se, int qs, int qe, int
 si)
{
    // If segment of this node is a part of given range, th
en return
    // the sum of the segment
    if (qs <= ss && qe >= se)
        return st[si];

    // If segment of this node is outside the given range
    if (se < qs || ss > qe)
        return 0;

    // If a part of this segment overlaps with the given ra
```

All Categories

Select Category

Recent Comments

Rashmi Mishra check this solution...

Check if two given strings are at edit distance one · 1 hour ago

Rashmi Mishra sorry my bad

Reverse an array without affecting special characters · 1 hour ago

Rashmi Mishra I doubt that this question can be solved...

Check if two given strings are at edit distance one · 1 hour ago

Ambuj Mani Tripathi GeeksforGeeks what is the need of 4th Argument...

Find a peak element · 2 hours ago

Pratyush Singh the best book for beginner to learning C

C · 2 hours ago

```
nge
    int mid = getMid(ss, se);
    return getSumUtil(st, ss, mid, qs, qe, 2*si+1) +
           getSumUtil(st, mid+1, se, qs, qe, 2*si+2);
}

/* A recursive function to update the nodes which have the
given
   index in their range. The following are parameters
    st, si, ss and se are same as getSumUtil()
    i    --> index of the element to be updated. This index
 is
            in input array.
   diff --> Value to be added to all nodes which have i in
range */
void updateValueUtil(int *st, int ss, int se, int i, int di
ff, int si)
{
    // Base Case: If the input index lies outside the range
 of
    // this segment
    if (i < ss || i > se)
        return;

    // If the input index is in range of this node, then up
date
    // the value of the node and its children
    st[si] = st[si] + diff;
    if (se != ss)
    {
        int mid = getMid(ss, se);
```

Aditya Siddheshwar Upadhyay if i have given a string as an input. then how...

Rearrange a string so that all same characters become at least d distance away · 2 hours ago

Tags

Adobe Advance Data Structures Advanced Data Structures Amazon array Backtracking Bharti SoftBank (HIKE) Bit Magic C++ CN c puzzle D-E-Shaw DBMS Divide and Conquer Dynamic Programming Flipkart GATE GATE-CS-2012 GATE-CS-C-Language GATE-CS-DS-&-Algo GATE-CS-Older GFacts Goldman Sachs Google Graph Greedy Algorithm Hashing Interview Experience Java MAQ Software MathematicalAlgo Matrix Microsoft Morgan Stanley Operating systems

```
        updateValueUtil(st, ss, mid, i, diff, 2*si + 1);
        updateValueUtil(st, mid+1, se, i, diff, 2*si + 2);
    }
}

// The function to update a value in input array and segmen
t tree.
// It uses updateValueUtil() to update the value in segment
 tree
void updateValue(int arr[], int *st, int n, int i, int new_
val)
{
    // Check for erroneous input index
    if (i < 0 || i > n-1)
    {
        printf("Invalid Input");
        return;
    }

    // Get the difference between new value and old value
    int diff = new_val - arr[i];

    // Update the value in array
    arr[i] = new_val;

    // Update the values of nodes in segment tree
    updateValueUtil(st, 0, n-1, i, diff, 0);
}

// Return sum of elements in range from index qs (quey star
t)
```

```
// to qe (query end).  It mainly uses getSumUtil()
int getSum(int *st, int n, int qs, int qe)
{
    // Check for erroneous input values
    if (qs < 0 || qe > n-1 || qs > qe)
    {
        printf("Invalid Input");
        return -1;
    }

    return getSumUtil(st, 0, n-1, qs, qe, 0);
}

// A recursive function that constructs Segment Tree for ar
ray[ss..se].
// si is index of current node in segment tree st
int constructSTUtil(int arr[], int ss, int se, int *st, int
 si)
{
    // If there is one element in array, store it in curren
t node of
    // segment tree and return
    if (ss == se)
    {
        st[si] = arr[ss];
        return arr[ss];
    }

    // If there are more than one elements, then recur for
left and
    // right subtrees and store the sum of values in this n
```

```
ode
    int mid = getMid(ss, se);
    st[si] =  constructSTUtil(arr, ss, mid, st, si*2+1) +
              constructSTUtil(arr, mid+1, se, st, si*2+2);
    return st[si];
}

/* Function to construct segment tree from given array. Thi
s function
   allocates memory for segment tree and calls constructSTU
til() to
   fill the allocated memory */
int *constructST(int arr[], int n)
{
    // Allocate memory for segment tree

    //Height of segment tree
    int x = (int)(ceil(log2(n)));

    //Maximum size of segment tree
    int max_size = 2*(int)pow(2, x) - 1;

    // Allocate memory
    int *st = new int[max_size];

    // Fill the allocated memory st
    constructSTUtil(arr, 0, n-1, st, 0);

    // Return the constructed segment tree
    return st;
}
```

```
// Driver program to test above functions
int main()
{
    int arr[] = {1, 3, 5, 7, 9, 11};
    int n = sizeof(arr)/sizeof(arr[0]);

    // Build segment tree from given array
    int *st = constructST(arr, n);

    // Print sum of values in array from index 1 to 3
    printf("Sum of values in given range = %d\n",
           getSum(st, n, 1, 3));

    // Update: set arr[1] = 10 and update corresponding
    // segment tree nodes
    updateValue(arr, st, n, 1, 10);

    // Find sum after the value is updated
    printf("Updated sum of values in given range = %d\n",
            getSum(st, n, 1, 3));
    return 0;
}
```

Output:

```
Sum of values in given range = 15
Updated sum of values in given range = 22
```

Time Complexity:

Time Complexity for tree construction is O(n). There are total 2n-1 nodes, and value of every node is calculated only once in tree construction.

Time complexity to query is O(Logn). To query a sum, we process at most four nodes at every level and number of levels is O(Logn).

The time complexity of update is also O(Logn). To update a leaf value, we process one node at every level and number of levels is O(Logn).

Segment Tree | Set 2 (Range Minimum Query)

References:

http://www.cse.iitk.ac.in/users/aca/lop12/slides/06.pdf

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

117 Comments Category: Advanced Data Structure Tags: Advance Data Structures, Advanced Data Structures, SegmentTree

Related Posts:

- Centroid Decomposition of Tree
- Gomory-Hu Tree | Set 1 (Introduction)
- kasai's Algorithm for Construction of LCP array from Suffix Array
- Overview of Data Structures | Set 3 (Graph, Trie, Segment Tree and Suffix Tree)
- Heavy Light Decomposition | Set 2 (Implementation)
- Heavy Light Decomposition | Set 1 (Introduction)
- Count Inversions of size three in a give array

- Count inversions in an array | Set 3 (Using BIT)

Previous post in category Next post in category

117 Comments GeeksforGeeks

wilson • a month ago

https://ideone.com/sx61st

linked representation (tree)

hazara mullah • a month ago

http://math.stackexchange.com/...

http://stackoverflow.com/quest...

• Reply • Share ›

Shahil Sabbag • 2 months ago

Please tell how to insert a element in segment tree.i.e when array

• Reply • Share ›

Gurpreet Singh • 2 months ago

this array representation of ST will eat up a lot of space, isn't there

• Reply • Share ›

haikent • 3 months ago

Python

http://haikent.blogspot.in/201...

• Reply • Share ›

Aditi Paul • 4 months ago

The max_size can be calculated by 2*n-1. Exponential calculation

2 • Reply • Share ›

Satish ➚ Aditi Paul • 3 months ago

Not true. The size of the segment tree should always be (2

construct the segment array of size 15. ([Next greater pow
segment tree as a perfect binary tree. Else, the implement
the extra scenarios.

2 ^ | v • Reply • Share ›

Mostafa Hany Gomaa ➚ Aditi Paul • 4 months ago

That's what I thought as well. I found that confusing.

^ | v • Reply • Share ›

Chanchana Sornsoontorn ➚ Mostafa Hany Gomaa • 3

To simplify this further use 4n for your max size an
is cheap. So you won't have to worry about miscal

^ | v • Reply • Share ›

Harsh Jain • 4 months ago

2^log2(n) = n. Don't understand why you are calculating the log an

^ | v • Reply • Share ›

Anonymous ➚ Harsh Jain • 3 months ago

Pay attention to the square brackets around log2(n). Thos
the n becomes 1 greater than power of 2, the value of log2
fraction). Although, there is just one addition to the size of
2*2^(log2(n)+1) - 1. In the above example, there are some
is not a power of 2, i.e. the children-space, of 5 and 11, are

^ | v • Reply • Share ›

They wanna get the next power of two. For example if you
tree size.

But the problem arise when you have n = 5, segment tree
You need segment tree of size (next power of two of 5)*2

And to calculate the next power of two, it's simple, log2(5)
and call it x. You do 2^x and that's it.

• Reply • Share ›

sam • 5 months ago

can anybody help me how complexity O(1) came for another solu
of all element till that index in some auxiliary array?????

• Reply • Share ›

Satish sam • 3 months ago

For building the sum arry, it takes O(n). But once you built
array.
Ex: arr->1,2,3,4,5; sumarry-> 1,3,6,10,15.
Now, if I query the sum from range 1 to 3, we return suma
general, for range[l,r], return summarry[r]-sumarry[l-1]. I ho

• Reply • Share ›

Falcon • 5 months ago

in my opinion this is very easy to understand instead of recursion

http://codeforces.com/blog/ent...

2 ^ | v • Reply • Share ›

Chanchana Sornsoontorn → Falcon • 3 months ago

For me, recursion is very self documentary code and it's s
efficient.

^ | v • Reply • Share ›

Satish → Falcon • 3 months ago

Awesome :) Thanks for sharing.

^ | v • Reply • Share ›

Sarthak Munshi • 5 months ago

If i have a range L..R . I need sum of all the elements in that range
.....n*a[R] ? How can i get this ?

^ | v • Reply • Share ›

gagan nagpal • 6 months ago

For the " sum of given range" question, is it the fastest known algo

^ | v • Reply • Share ›

Billionaire • 7 months ago

In `void updateValueUtil(int *st, int ss, int se, int i, int diff, int index)`
`si` to replace `index` so that it is obviously that `si` means Segme

1 ^ | v • Reply • Share ›

GeeksforGeeks Mod → Billionaire • 6 months ago

Thanks for the suggestion. We have updated the variable n

Rahul Ranjan • 7 months ago

Height of segment tree is given wrong----

For a binary tree with number of nodes N, we have height as log[N
but here total nodes, N is 2n-1 where n is size of array.....
So putting this value in the formula, we have
log[2n-1+1] = log[2n] = 1+logn

Reply • Share ›

cdCoding ↗ Rahul Ranjan • 4 months ago

It is the same as the program calculated

Reply • Share ›

Vivek Garg • 7 months ago

Another implementation is here :
http://gargvivekcse12.blogspot...

Reply • Share ›

codemonk ↗ Vivek Garg • 7 months ago

i think in newNode function you should also set left and rigl

Reply • Share ›

Vivek Garg ↗ codemonk • 6 months ago

ya sorry ! My bad . I updated it.

Reply • Share ›

Abhiroj Panwar • 8 months ago

if tree construction complexity is O(n)..then why to use segment t

1 • Reply • Share ›

Waquar ↗ Abhiroj Panwar • 7 months ago

When there are lot of query and update operations segme

• Reply • Share ›

Prashant Singh • 8 months ago

Please make an article on lazy propagation.

• Reply • Share ›

Rini • 9 months ago

http://ideone.com/f6P3aD

• Reply • Share ›

vergil • 9 months ago

can u make an article on lazy propogation...basic segmentation tr programming....

1 • Reply • Share ›

vergil ↗ vergil • 9 months ago

i think the complexity would be O(N) in case we want to up such we'll need lazy propogation...

• Reply • Share ›

constructSTutil returns int value but construct ST doesnot have a

Reply • Share ›

Ashish Ayushi Srivastava • 9 months ago

because construct ST doesn't need the value returned by

Reply • Share ›

Vinod • 9 months ago

Iterative simple 60 line of code with proper comments and test ca

http://ideone.com/zWJ0pq

1 Reply • Share ›

vergil • 9 months ago

e-maxx.ru....best explanation...with minimum code...

Reply • Share ›

Vivek Dadu • 10 months ago

And here is a more detailed and more intuitive blog on segment tr

Reply • Share ›

Mission Peace • a year ago

My video on segment tree

https://www.youtube.com/watch?...

10 Reply • Share ›

Krishna Mission Peace • 8 months ago

Great explanation Tushar, I am little confused on few thing
be really helpful if you could help me out of my confusions.
1. Create Segment tree: How at max the size of the Segm
times N, N being the total elements in the provided array ?
elements are 5 so we will need 15 as the size of the Segm
2. Search min in range : How in worst case it looks in 4 dif
directions ? For a given range we find if its Partial Overlap
choose to either have a look at both the Nodes other wise
very large number in case of No overlap or the value at no
Full overlap).

• Reply • Share ›

Chanchana Sornsoontorn ➚ Krishna • 3 months ago

4 is the approximation in case you are 'lazy' to do t
size. This approximation is bigger than the needed
more simple to use 4n as the size.

Now, where does 4 come from?
Suppose you have n = 8, you can use size 16
But if you have n = 9, you cannot use size 16 for yo
approximately 4*9 = 36, so you can create array of

• Reply • Share ›

Shashank Kumar ➚ Mission Peace • 10 months ago

+1 for clear explanation.
Maybe this video should be made part of the article.

• Reply • Share ›

dmr ➚ Mission Peace • a year ago

Thanks dude...very nice!!

• Reply • Share ›

Sumit Vohra • a year ago

i made a tree just as the segment tree shown above and did the le
the array is that fine
http://ideone.com/e3dxTq

• Reply • Share ›

Mr. Lazy • a year ago

Learned Something New! :)

• Reply • Share ›

O • a year ago

I'm not sure I understand the time complexity for query operation.
appears to be O(n). You mentioned that it is O(log n) because we
please explain how you came to this number? I checked the IIT lin
seems to lack any explanation of this point.

• Reply • Share ›

mauricepatel37 ➚ O • 10 months ago

It is O(log n) since you already have sum values stored in
at every level to see whether the range for query is in the g
At the worst case you need to traverse from root to bottom
tree which is (log n).

Hence complexity is O(log n)

Reply • Share ›

sreekanth ➚ mauricepatel37 • 8 months ago

In the given example if you want to find the sum for how it become O(log n) solution.

Reply • Share ›

sandeep • a year ago

could u make a post on lazy propagation

Reply • Share ›

guest • a year ago

while solving a problem I required segment tree of size 3*n , if size was n..

Could anyone please explain me why it requires size more than 2

Reply • Share ›

Guest • a year ago

sdf

1 Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy