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
2 Change value of a specified element of the array arr[i] = x where 0 <= i <= n-1.

A **simple solution** is to run a loop from l to r and calculate sum of elements in given range. To update a value, simply do arr[i] = x. The first operation takes O(n) time and second operation takes O(1) time.

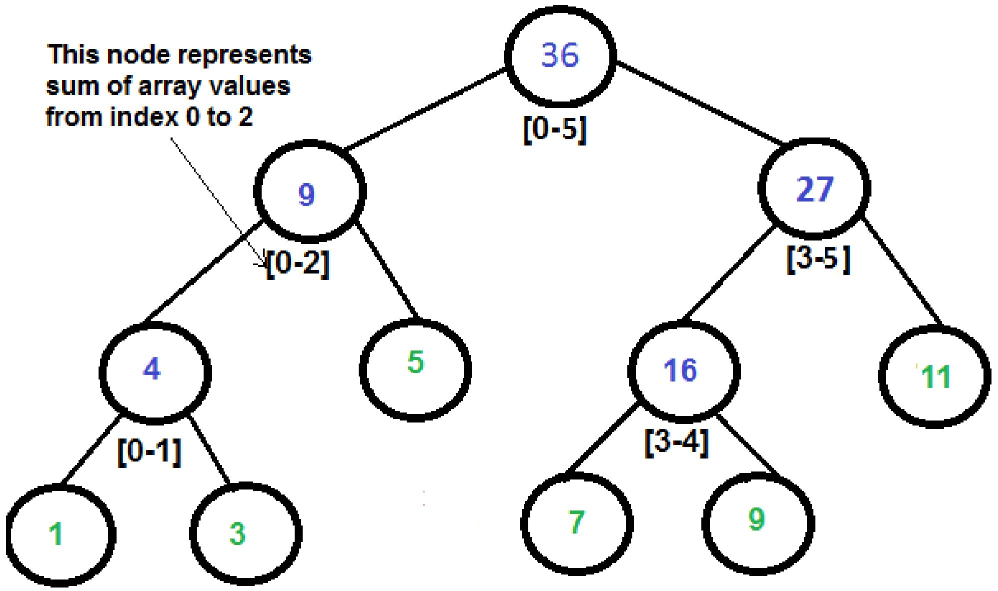
Another solution is to create another array and store sum from start to i at the ith index in this array. Sum of a given range can now be calculated in O(1) time, but update operation takes O(n) time now. This works well if the number of query operations are large and very few updates.

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 $\lfloor (i-1)/2 \rfloor$.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms



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 Full Binary Tree because we always divide segments in 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)
{
```

Popular Posts
All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
   if range of node is within l and r
        return value in node
   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 *diff* be the value to be added. We 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.

```
// Program to show segment tree operations like construction, query an
#include <stdio.h>
#include <math.h>

// A utility function to get the middle index from corner indexes.
int getMid(int s, int e) {  return s + (e -s)/2;  }

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

    st    --> Pointer to segment tree
    index --> Index of current node in the segment tree. Initially 0 i
             passed as root is always at index 0
    ss & se  --> Starting and ending indexes of the segment represente
                 current node, i.e., st[index]
    qs & qe  --> Starting and ending indexes of query range */
int getSumUtil(int *st, int ss, int se, int qs, int qe, int index)
{
    // If segment of this node is a part of given range, then return t
    // sum of the segment
    if (qs <= ss && qe >= se)
        return st[index];
    // 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 range
    int mid = getMid(ss, se);
    return getSumUtil(st, ss, mid, qs, qe, 2*index+1) +
           getSumUtil(st, mid+1, se, qs, qe, 2*index+2);
}

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

    // If the input index is in range of this node, then update the va
    // of the node and its children
    st[index] = st[index] + diff;
    if (se != ss)
    {
        int mid = getMid(ss, se);
        updateValueUtil(st, ss, mid, i, diff, 2*index + 1);
        updateValueUtil(st, mid+1, se, i, diff, 2*index + 2);
    }
}

// The function to update a value in input array and segment 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 start) 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 array[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 current 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 node
    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. This function
   allocates memory for segment tree and calls constructSTUtil() to
   fill the allocated memory */
int *constructST(int arr[], int n)
{
    // Allocate memory for segment tree
    int x = (int)(ceil(log2(n))); //Height of segment tree
    int max_size = 2*(int)pow(2, x) - 1; //Maximum size of segment tre
    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).

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.

HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.

LexisNexis Learn More

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree

45 Tweet 4 1

Writing code in comment? Please use **ideone.com** and share the link here.

44 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion…

prashant • 3 days ago

void update(tnode* root,int arr[],int low,int high,int ind,int val)

{

if(low==high)

{

root->data=val;

return;

}

int mid=(low+high)/2;

if(ind>mid)

update(root->rchild,arr,mid+1,high,ind,val);

if(ind<=mid)

see more

^ | ˅ • Reply • Share ›

prashant • 3 days ago

note the crucial points

1-segment tree is not complete binary tree

2-the left and right subtree are divided based on middle values

3-the sum and update code are similar to bianry search code so its implemen

for update process

construction is simple ...just recursively go down dividing the index into 2 halve

node data and during unwinding phase adjust root->data

for range sum

if ind1>mid then just move to right subtree

if ind2<=mid move to left

else split the range index into 2 halves

```
int sum(tnode* root,int low,int high,int l,int h)

{

if((low==l)&&(high==h))

return root->data;
```

see more

Reply • Share ›

dmr • 20 days ago

We can use segment trees for finding sum in a given range. But how can this in a given range in O(logn) time? This is a question in References link (http://www.cse.iitk.ac.in/user... given above.

Reply • Share ›

prshant jha • 2 months ago

here is my update version in 0(logn) complexity with much simpler implement http://ideone.com/SppdWT

2 ^ | ⌄ • Reply • Share ›

GOPI GOPINATH ➦ prshant jha • a month ago

will the complexity in your implementation be O(logn) ???? but to find tl
runtime ryt ????

^ | ⌄ • Reply • Share ›

prashant jha • 2 months ago

#include<iostream>

using namespace std;

struct node

{

node* lchild;

int data;

node* rchild;

node()

{

lchild=NULL;

rchild=NULL;

see more

^ | ⌄ • Reply • Share ›

prashant jha • 2 months ago

here is the implementation of the segment tree
http://ideone.com/JPDzqz

• Reply • Share ›

nwoebcke • 2 months ago

Although this puts an additional O(log(n)) of memory on the heap, I think it mal
recursion a little easier to follow. Also it is C++. To preserve the C language, y
classes and rename the methods to functions with a struct pointer parameter

```
class Range {
  public:
   int start;
   int end;
   Range(s, e) : start(s), end(e) {}
   void init(s, e) {
      start = s;
      end = e;
   }
   inline bool isInside(Range *other) {
      return start >= other->start && end <= other->end;
   }
   inline bool isOutside(Range *other) {
      return start > other->end || end < other->start;
```

see more

• Reply • Share ›

Puneet Jaiswal • 3 months ago

Would this work as tree implementation

```
public class SegmentTree {
```

```
int leftIndex;
int rightIndex;
int sum;
STNode leftNode;
STNode rightNode;
}

static STNode constructSegmentTree(int[] A, int l, int r) {
if (l == r) {
STNode node = new STNode();
node.leftIndex = l;
node.rightIndex = r;
node.sum = A[l];
```

see more

Reply • Share ›

mallard • 4 months ago

sorry but i think the language of implementation of above code is C and i think getting error when i am trying to run this code.I am using codeblock.

Reply • Share ›

Newbie90 • 4 months ago

What is the difference between the constructST and constructSTUtil functions constructSTUTil fucntion.

Reply • Share ›

Ankur Sao → Newbie90 • 7 days ago

ConstructST only allocates memory for the segment tree array and ca actually fills up the segment tree array. ConstructST than returns this

Reply • Share ›

Vu Duc Minh • 4 months ago

I do not think the procedure "updateValueUtil" is a good one (even it is correct)
should update from a leaf to the root; like the "constructSTUtil" procedure. In fa
"constructSTUtil". We only need one procedure for all two tasks.

^ | ˅ • Reply • Share ›

Adrian Carballo • 4 months ago

Hey, great tutorial, I wrote a python implementation here https://github.com/adı

^ | ˅ • Reply • Share ›

Avinash Ks • 6 months ago

Just one doubt, in SumUtil, isn't qs supposed to be greater than ss and qe les
of ss - se

^ | ˅ • Reply • Share ›

Denis • 8 months ago

Hi, you wrote : "size of memory allocated for segment tree will be 2*2^|log2n|-1
where I assume n is a number of leafs in the tree. This is seems not to
be true using your example of a segment tree : "{1,3,5,7,9,11}",
where n=6. Thus using 2*2^|log2n|-1 size of memory allocated is 7, which is n
are 11 nodes. Suppose n should be replaced
on the (2*n-1) in your expression.

^ | ˅ • Reply • Share ›

Denis ➜ Denis • 8 months ago

Sorry, I've got it. You are using ceil() function in this case. So 2*2^ceil(l
tree size. I was thinking about floor() instead of ceil().

^ | ˅ • Reply • Share ›

Denis • 8 months ago

Hi, you wrote : "size of memory allocated for segment tree will be ", where I as

tree. This is seems not to be true using your example of a segment tree : "{1,3
where n=6. Thus using size of memory allocated is 7, which is not true becau
Suppose n should be replaced on the (2*n-1) in your expression.

• Reply • Share ›

denial • 9 months ago

@geeksforgeeks:
Change suggestion in the paragraph "Query for Sum of given range". It should

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

   if range of node is completely outside l and r
        return 0

    return getSum(node's left child, l, r) +
           getSum(node's right child, l, r)
}
```

let me know if I'm wrong. :)

• Reply • Share ›

denial • 9 months ago

@geeksforgeeks
Change suggestion in the paragraph "Query for Sum of given range" above :

You written it as :

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

should be changed to this:

see more

^ | ⌄ • Reply • Share ›

Prakhar Jain • 10 months ago
Time Complexity of query is O(log n) because we process at most "4 nodes"
nodes" which is wrong. For example take range [1-3] in your example and mal
function, you will see there are at most 4 nodes at each level.
Even it is also given in the iitk link you have given at the end.

```
  /* Paste your code here (You may delete these lines if not writing c
```

1 ^ | ⌄ • Reply • Share ›

GeeksforGeeks → Prakhar Jain • 10 months ago
@Prakhar Jain: Thanks for pointing this out. we have updated the post

^ | ⌄ • Reply • Share ›

Prakhar Jain → GeeksforGeeks • 10 months ago
Also, to update a leaf we process "two nodes" at each level, no

```
/* Paste your code here (You may delete these lines if r
```

^ | ˅ • Reply • Share ›

kartik • 11 months ago

How do we do the updation if we have to update more than 2 values

like we have to increase all number in range a to b by 2

how our update function do this in O(log(n))

can any body plz help

```
/* Paste your code here (You may delete these lines if not writing coo
```

^ | ˅ • Reply • Share ›

Sumanth Bandi • 11 months ago

&#039&#039Since the tree is represented using array and relation between pa
maintained, size of memory allocated for segment tree will be 2*(2^ceil(log2n)
line.

Why not just 2*n - 1? What are the bad sequences of just allotting 2*n - 1 nod
Anybody pls help..

^ | ˅ • Reply • Share ›

sumanth232 • 11 months ago

Since the tree is represented using array and relation between parent and chil
memory allocated for segment tree will be 2*(2^ceil(log2n)) - 1.

Why not just 2*n - 1 ? What are the bad sequences of just allotting 2*n - 1 noc

Anybody pls help..

^ | ⌄ • Reply • Share ›

alveko ➦ sumanth232 • 11 months ago

The array must have enough elements to include a possible right-mos
most leaf increases with a step of power of 2. The size of (2*n-1) migh

```
// segment tree size (n is the number of elements in the input
//     (log2ceil(n))   is the level that can hold all distinct
//   2^(log2ceil(n))   is the number of elements at that level
// 2*2^(log2ceil(n))-1 is the total number of elements in the t
```

/ Alexander K.

^ | ⌄ • Reply • Share ›

sumanth232 ➦ alveko • 11 months ago

Thanks.. that made it clear..

```
/* Paste your code here (You may delete these lines if r
```

^ | ⌄ • Reply • Share ›

abhishek08aug • 11 months ago

Intelligent :D

^ | ⌄ • Reply • Share ›

Ouditchya Sinha • a year ago

Very nicely explained! Thank You GeeksforGeeks... :) Just one question, wher
the value in array & construct another segment tree? It seems to be working h

^ | ⌄ • Reply • Share ›

prasad • a year ago
I have difficulty understanding the time complexity of FindMin().

At each node we are splitting the problem in to two sub problems of equal size

T(n) = 2T(n/2) + 1;

I think this reduces to O(n).

Please correct me if I am wrong any where?

^ | ⌄ • Reply • Share ›

Abhay • a year ago
I think in updateValueUti function it should be
updateValueUtil(st, ss, mid, i, diff, 2*index + 1) instead of
updateValueUtil(st, 0, mid, i, diff, 2*index + 1)

^ | ⌄ • Reply • Share ›

GeeksforGeeks ➜ Abhay • a year ago
Thanks for pointing this out. We have updated the code.

^ | ⌄ • Reply • Share ›

Gaurav Jain • a year ago
This is incorrect...
Let us reconsider the example of array 1,3,5,7,9,11

if i have to calculate sum of indices 2 to 4. This should be 5+7+9=21. But usin
calculate the same.

Segment tree will work only if the indices are in first half or 2nd half, but when
not work

1 ^ | ⌄ • Reply • Share ›

GeeksforGeeks ➦ Gaurav Jain • a year ago
We ran the above given code for your input and it produced the correct
^ | ⌄ • Reply • Share ›

Sitesh • a year ago
BIT is more efficient in this case. Relatively faster than Segment trees, Lesser

Time complexities : O(log N)
Space complexities: O(N)
More details here: http://www.algorithmist.com/in...

[sourcecode language="C++"]
#include <vector>
using namespace std;

// In this implementation, the tree is represented by a vector<int>.
// Elements are numbered by 0, 1, ..., n-1.
// tree[i] is sum of elements with indexes i&(i+1)..i, inclusive.

// Creates a zero-initialized Fenwick tree for n elements.
vector<int> create(int n) { return vector<int>(n, 0); }

// Returns sum of elements with indexes a..b, inclusive
int query(const vector<int> &tree, int a, int b) {

see more

^ | ⌄ • Reply • Share ›

Kumar • a year ago
Nice explanation. But there's another tree structure which is precisely meant t
called Binary indexed trees, which is simpler, powerful and easy to maintain.

Segment trees are really good for answering range minimum queries & interva

Here's a complete working implementation of segment tree in C#

[sourcecode language="C#"]
/* Paste your code here (You may delete these lines if not writing code)

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Algorithms.Trees.Base;

namespace Algorithms.Trees
{
public class SegmentTree<K> where K:IComparable<K>

see more

Reply • Share ›

kT • a year ago

```
Hi,
I think this is incorrect.
constructSTUtil(arr, 0, n-1, st, 0);
 // Return the constructed segment tree
    return st;



This should do the job.
st = constructSTUtil(arr, 0, n-1, st, 0);
```

```
Please correct me else.

Thanks
```

Reply • Share ›

GeeksforGeeks ➜ kT • a year ago

Please take a closer look at the code. The recursive function construct
sum of leaf nodes under it). st is a pointer to the constructed segment

Reply • Share ›

kT ➜ GeeksforGeeks • a year ago

Ok, yes. My bad!

Reply • Share ›

Vikas • a year ago

May you please give the code to implement segment tree to store intervals ?

Reply • Share ›

sreeram ➜ Vikas • a year ago

Yeah ..can you please provide that implementation also ?

Reply • Share ›

Venki • a year ago

You are using 0th location also, so, left child is at 2*i+1 and right child at 2*i+2
parent is at i/2.

Power function can be excluded with simple shift operation.

Reply • Share ›

GeeksforGeeks ➜ Venki • a year ago

@Venki: Thanks for pointing this out. The line of explanation was for st

index 0. We have updated the explanation to match with code. The coc
same.

• Reply • Share ›

Subscribe Add Disqus to your site