**Segment Tree**

1. **Definition of Segment Tree**

Imagine that you have an array A: A[0], A[1],…, A[n-1] (n is the number elements of the array). You want to do these operations with the array: find sum of the elements which from index l to index r in the array (A[l] + A[l+1] + … + A[r-1] + A[r], ), find maximum value, minimum value of elements in the range [l...r] of the array, update value in the rang [l…r] (update one elements, update all elements such as plus a value x to all elements in the range). With all these operations, the time complexity is .

There is one simple solution that you can run some loops with the whole arrays to do these operations. But it’s not a effective way because the time complexity is

So, there is another solution that is more effective than the normal way which uses some loops though the code-implement is more complicated. That is **Segment Tree**. A Segment Tree is a tree data structure. Specifically, this tree’s nodes has 2 cases. First, the nodes is the leaf of the tree. Second, each node has 2 children. There is no case that a node has **one** child.

This kind of tree has many applications in competitive coding challenge by dealing with sequence of numbers.

1. **Implement Segment Tree**

The array **A** has **n** elements. **l** is the left index, **r** is the right index in the array, **l** and **r** form a segment in the array, .

The first node (root) of Segment Tree represents for information of all elements in the array (range: [1…n]). The leaf nodes represent for elements of the input array (A[0], A[1], …, A[n-1]). Each internal node represents for some merging of the leaf nodes. If a node represents for information in the range [l…r], then its children will represent for information in the range [l…(l+r)/2] and [(l+r)/2 + 1 … r].

Example with array A has 7 elements {0,1,2,3,4,5,6}, Segment tree will be implemented:

A picture containing text, pool ball, night sky

Description automatically generated

In coding, we can easy implement Segment Tree by using 1D array. The first element A[0] represents for root node. The i element will have 2 children is i\*2 (left child) and i\*2+1 (right child). The memory used for Segment Tree will be no more than 4\*n elements.

1. **Use Segment Tree to deal with problems**

**Example 1:** Find sum of the elements in the array A which in range [l…r]

A = {5,8,6,3,2,7,2,6}, l = 2; r = 7.

Background pattern

Description automatically generatedSegment Tree of this problem:

The leaf nodes are the elements in the array A (5,8,6,3,2,7,2,6).

The upper nodes are the sum values of the leaf node: 13 = 5 + 6, 9 = 6 + 3, 9 = 2 + 7, 8 = 2 + 6, the next upper nodes and the root node are the same. So, the root node is the sum of all elements in the array.

Now, we need to calculate the sum: A[2] + A[3] + … + A[7]

We can use a recursive function to solve this problem, pseudo-code:



These steps are explaination of the code above:

* Start function at root node ([0…7])

Because [2…7] range has a small part in [0…7] range, continue to check node-> left ([0…3]) and node->right ([4…7])

* Call function at node [0…3]

Because [2…7] has a small part in [0…3] range, continue to check node->left ([0…1]) and node->right([2…3])

* Call function at node [0…1]

Because [2…7] out of range [0…1] so return 0.

* Call function at node [2…3]

Because [2…7] has the range [2…3] so return its value: 9

Now return result of function at node [0…3]: 0 + 9 = 9.

* Call function at node [4…7]

Because [2…7] has the range [4...7] so return its value: 17

Now return of function at node [0…7]: 9 + 17 = 26.

* Now the function ends, the final result is 26. (6+3+2+7+2+6 = 26)

C/C++ code create Segment Tree in this problem:



C/C++ code getSum function:



**Example 2:** Find minimum value through the elements in the array A which in range [l…r]

A = {5,8,6,3,2,7,2,6}, l = 3; r = 7.

Background pattern

Description automatically generated

The upper nodes is the smaller node of its children. (5 < 8 => 5, 3 < 6 => 3, ..). Now we need to find the smallest element in range A[3], A[4], .. A[7].

We can use a recursive function to solve this problem, pseudo-code:



These steps are explaination of the code above:

* Start function at root node ([0…7])

Because [3…7] range has a small part in [0…7] range, continue to check node-> left ([0…3]) and node->right ([4…7])

* Call function at node [0…3]

Because [3…7] has a small part in [0…3] range, continue to check node->left ([0…1]) and node->right([2…3])

* Call function at node [0…1]

Because [3…7] out of range [0…1] so return NOTHING.

* Call function at node [2…3]

Because [3…7] has a part in the range [2…3] so continue to check node->left ([2]) and node->right([3])

[2] doesn’t in [3…7] so return NOTHING.

[3] is in [3…7] so return its value: 3.

Now return of function at node [0…3] is 3.

* Call function at node [4…7]

Because [3…7] has the range [4...7] so return its value: 2

Now return of function at node [0…7]: 2 < 3 => Min = 2

* Now the function ends, the final result is 2.

C/C++ code create Segment Tree in this problem (Min(a,b) return smaller value between a and b)



C/C++ code getSum function:



**Update a element’s value by index (pos) in the array A.**

Just like other operations with Segment Tree. Updating function is taken by recursive function. The input of the function is an index (pos) and a new value of that element. With Segment Tree, we must go from the root node and update all node that include the index (pos) node.

C/C++ code update an element:



Red line: Depend on the operation that the problem deal with.

Example problem in ex1: Calculate the sum

tree[v] = tree[v\*2+1] + tree[v\*2+2] // Sum of two children

Example problem in ex2: Find minimun value

tree[v] = Min(tree[v\*2+1], tree[v\*2+2])

// Find smaller value between two children