Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITL502 **DATE:14/11/22**
COURSE NAME: Advanced Data Structures Laboratory **CLASS/Div: A**
SAP ID:60003200076 **Academic Year 2022-23**

EXPERIMENT NO. 5

CO/LO: LO1

AIM: To Implement Segment tree.

DESCRIPTION OF EXPERIMENT:

Segment Tree is a basically a binary tree used for storing the intervals or segments. Each node in the Segment Tree represents an interval Consider an array A of size N and a corresponding Segment Tree T

- The root of T will represent the whole array A0N-1
- Each leaf in the Segment Tree T will represent a single element A[i] such that 0≤1<N.
- The internal nodes in the Segment Tree represents the union of elementary intervalsA[I:j] where 0 ≤1< N.

The root of the Segment Tree represents the whole array A[0:N-1]. Then it is broken down into two half intervals or segments and the two children of the root in turn represent the A[0:(N-1)/2] and A[(N-1)/2+1: (N-1)]. So in each step, the segment is divided into half and the two children represent those two halves. So the height of the segment tree will be log2N. There are N leaves representing the N elements of the array. The number of internal nodes isN-1 so a total number of nodes are 2*N-1.

Once the Segment Tree is built its structure cannot be changed. We can update the values of nodes but we cannot change its structure.

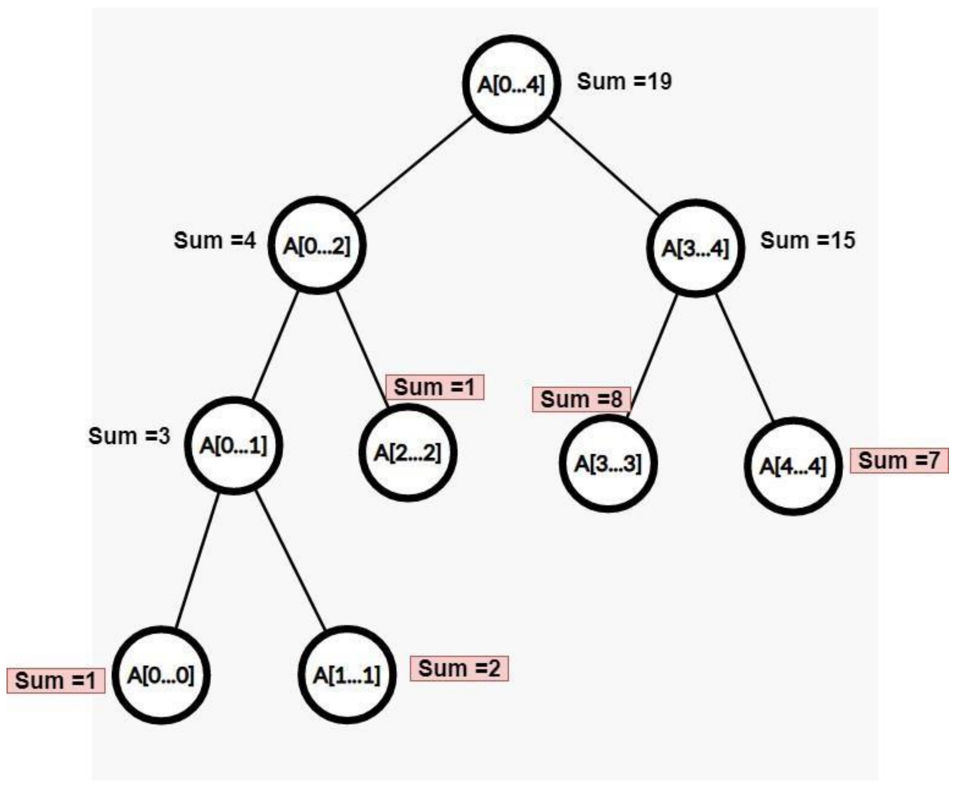
Building Tree: in this step, we create the structure of the segment tree variable and initializeit

Querying Tree: we may use this operation to run a range query on the array

In a Segment Tree, the array is stored at the leaves of the tree, while the internal nodes storeinformation about segments represented by its children. The internal nodes are formed in abottom-up manner by merging the information from its children nodes.

Segment Trees are useful whenever we're frequently working with ranges of numerical data.In this context, let's look at an example to better understand the Segment Tree by describing each step. We want to construct a Segment Tree array that allows us to find the sum of the elements in an array.

Consider an array A = [1,2,1,8,7]



- The root node is represented by A[0,n-1] which stores the sum of all the elements ofthe given array.
- Then the segment A[0,n-1] is divided into two halves, namely A[0,n/2] andA[n/2+1,n-1], and the sum of each half is computed and stored.

- Each of these two halves is further split into half, and the sum is computed and stored.
- This process continues until all the segments length becomes 1.
- So, at the first level, there is one node that is the root. In the second level, there aretwo nodes, third level four nodes,...until the number of nodes in a level becomes n.
- All the leaf nodes represent the individual elements of the array. As you can see in theabove segment tree, the highlighted sums are nothing but the element at that index.

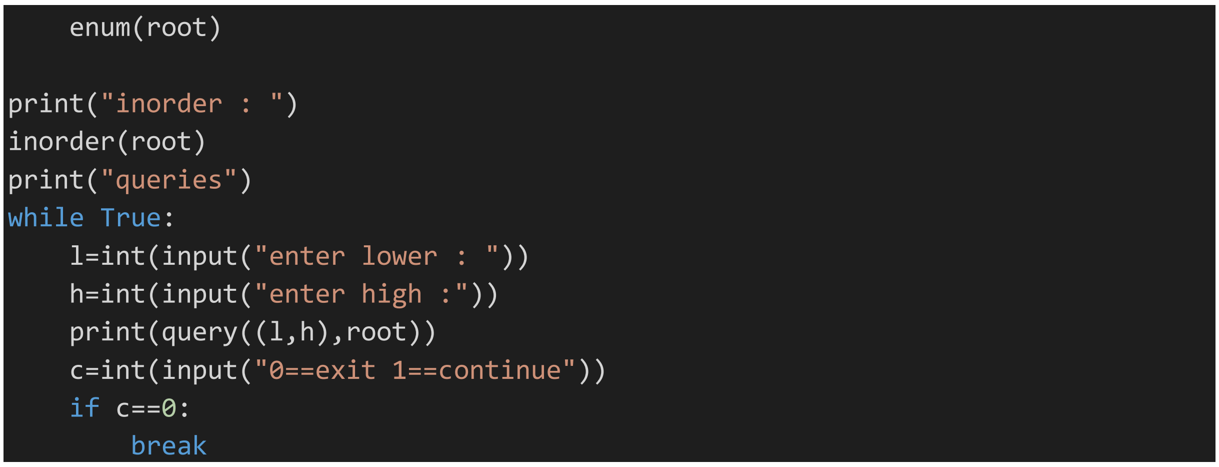
Properties of Segment Tree

- The height of the segment tree is O(log n) because while moving from the root to theleaves at every level, the length of the segments is reduced by half.
- The total number of nodes involved in the segment tree is 4*n.
- The total number of levels is log n and starting from one node at the first level, thenumber of nodes gets doubled at every level.
- So, total number of nodes = 1+2+4+8+... +2^(log n) = 2^(logn + 1) -1 < 4n.

Technology stack used: Python

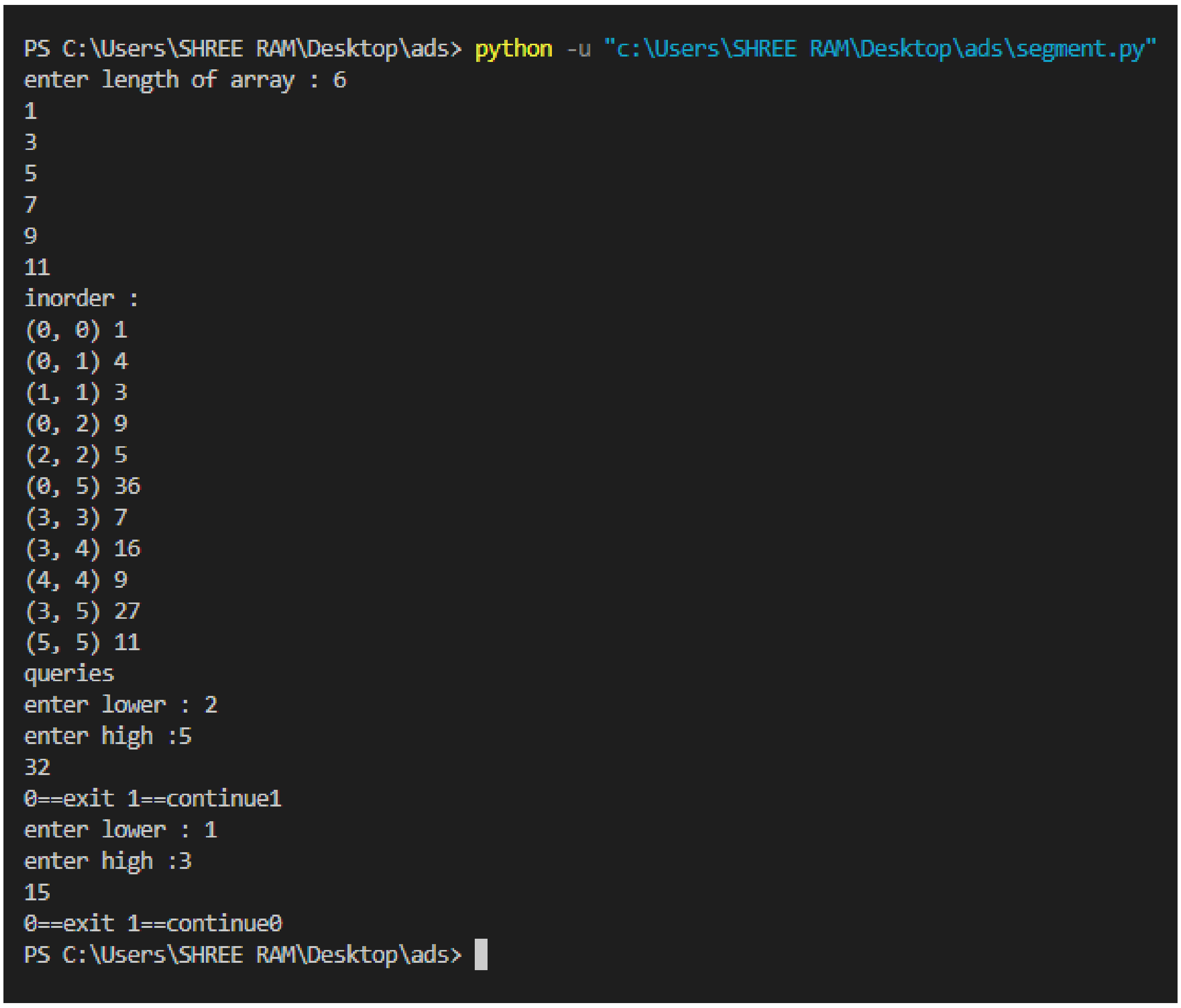
Program:

```
#segment tree
class node:
    def __init__(self,r):
        self.r=r
        self.left=None
        self.right=None
        self.val=False


def maketree(arr,root):
    if root.r[0]!=root.r[1]:
        #print(root.r[0],root.r[1])
        root.left=node((root.r[0],(root.r[0]+root.r[1])//2))
        root.right=node(((root.r[0]+root.r[1])//2+1,root.r[1]))
        maketree(arr,root.left)
        maketree(arr,root.right)
    elif(root.r[0]==root.r[1]):
        root.val=arr[root.r[0]]
        #print(root.r[0],root.r[1],root.val)

def enum(root):
    if root.val:
        a=0
    elif root.left.val and root.right.val:
        #print(root.r[0],root.r[1],root.val,root.left.val, root.right.val)
        root.val=root.left.val+root.right.val
    elif not root.val:
        #print(root.r[0],root.r[1],root.val,root.left.val, root.right.val)
        enum(root.left)
        enum(root.right)


def inorder(root):
    if root:
        inorder(root.left)
        print((root.r),root.val)
        inorder(root.right)

def query(interval,root):
    l=interval[0]
    h=interval[1]
    if l<=root.r[0] and h>=root.r[1]:
        return root.val
    elif (l>root.r[1] and h>root.r[1]) or (l<root.r[0] and h<root.r[0]):
        return 0
    else:
        return query(interval,root.left) + query(interval,root.right)



#arr=[1,3,5,7,9,11]
arr=[]
n=int(input("enter length of array : "))
for i in range(0,n):
    e=int(input())
    arr.append(e)

root=node((0,len(arr)-1))
maketree(arr,root)
while not root.val:
```



```
    enum(root)

print("inorder : ")
inorder(root)
print("queries")
while True:
    l=int(input("enter lower : "))
    h=int(input("enter high :"))
    print(query((l,h),root))
    c=int(input("0==exit 1==continue"))
    if c==0:
        break
```

<u>Output</u>:



```
PS C:\Users\SHREE RAM\Desktop\ads> python -u "c:\Users\SHREE RAM\Desktop\ads\segment.py"
enter length of array : 6
1
3
5
7
9
11
inorder :
(0, 0) 1
(0, 1) 4
(1, 1) 3
(0, 2) 9
(2, 2) 5
(0, 5) 36
(3, 3) 7
(3, 4) 16
(4, 4) 9
(3, 5) 27
(5, 5) 11
queries
enter lower : 2
enter high :5
32
0==exit 1==continue1
enter lower : 1
enter high :3
15
0==exit 1==continue0
PS C:\Users\SHREE RAM\Desktop\ads>
```

ANALYSIS:

Complexities:

The time complexity of tree construction is O(n), n being the number of elements put in thetree, as n calls will be made overall, as there are (2*n)-1 nodes in the tree and value of eachnode is computed only once during tree construction. The time complexity of a query operation is O($\log_2 n$) as a maximum of 4 nodes will be visited at any level of recursion andthe height of the tree is $\log_2 n$ which corresponds to the number of levels. The space complexity of a segment tree is O(n).

APPLICATIONS:

- Computational geometry.
- Geographic information systems.
- Machine learning.
- In its early days, the Segment Tree was used to efficiently list all pairs of intersectingrectangles from a list of rectangles in the plane.
- We can use this method to report the list of all rectilinear line segments in the planewhich intersect a query line segment.
- We use this technique to report the perimeter of a set of rectangles in the plane.
- More recently, the segment tree has become popular for use in pattern recognition andimage processing.
- Finding range sum/product, range max/min, prefix sum/product, etc
- Computational geometry
- Geographic information systems
- Static and Dynamic RMQ (Range Minimum Query)
- Storing segments in an arbitrary manner

Conclusion:

Thus we have implemented segment trees.