Binary Indexed Tree (BIT)

Aruneswari S **CB.EN.U4ECE18107**

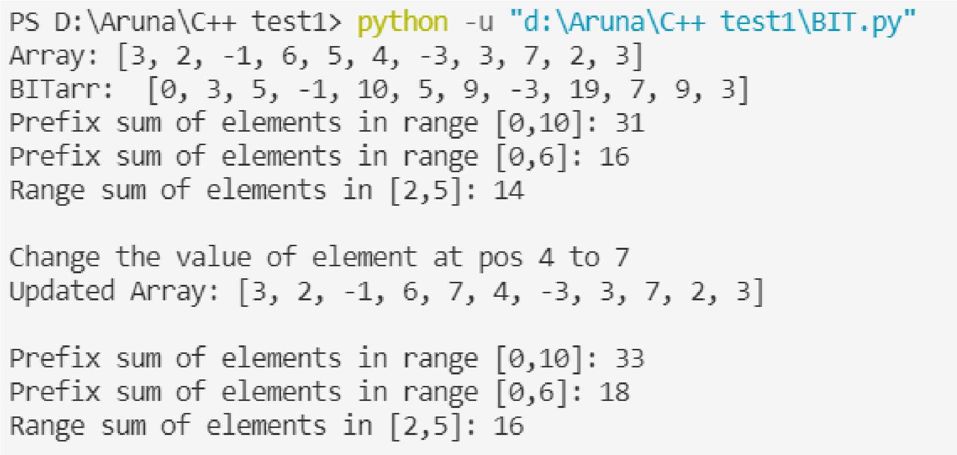
Python implementation of BIT data structure:

```python
class BIT:

    def __init__(self,arr,n):
        """"Initializing the BIT in O(n*log(n)) time"""
        self.size=n
        self.array=arr
        self.BITarr = [0]*(self.size+1)
        for index in range(len(self.array)):
            self.update(index,self.array[index])
        print("BITarr: ",self.BITarr)


    def sum_query(self, index):
        """Computing sum of elements in range [0,idx] """
        index += 1
        result = 0
        while index>0:
            result += self.BITarr[index]
            index -= index & (-index)
        return result

    def update(self, index, value):
        """Adding a value to the index-th element"""

        index += 1
        while index <= len(self.array):
            self.BITarr[index] += value
            index += index & (-index)


    def range_sum(self, start_index, end_index):
        """Computing the range sum [start_index, end_index]"""

        return self.sum_query(end_index) - self.sum_query(start_index - 1)
```

```
if _name__== '_main_':
    arr=[3,2,-1,6,5,4,-3,3,7,2,3]
    print("Array:",arr)
    bit = BIT(arr,len(arr))

    print("Prefix sum of elements in range [0,10]:", bit.sum_query(10))
    print("Prefix sum of elements in range [0,6]:", bit.sum_query(6))
    print("Range sum of elements in [2,5]:", bit.range_sum(2,5))
    print()
    bit.update(4, 2)
    print("Change the value of element at pos 4 to 7")
    new_array = [bit.range_sum(index, index) for index in range(len(arr))]
    print("Updated Array:",new_array)
    print()
    print("Prefix sum of elements in range [0,10]:", bit.sum_query(10))
    print("Prefix sum of elements in range [0,6]:", bit.sum_query(6))
    print("Range sum of elements in [2,5]:", bit.range_sum(2,5))
    print()
```

OUTPUT:

```
Array: [3, 2, -1, 6, 5, 4, -3, 3, 7, 2, 3]

BITarr:  [0, 3, 5, -1, 10, 5, 9, -3, 19, 7, 9, 3]

Prefix sum of elements in range [0,10]: 31

Prefix sum of elements in range [0,6]: 16

Range sum of elements in [2,5]: 14


Change the value of element at pos 4 to 7

Updated Array: [3, 2, -1, 6, 7, 4, -3, 3, 7, 2, 3]


Prefix sum of elements in range [0,10]: 33

Prefix sum of elements in range [0,6]: 18

Range sum of elements in [2,5]: 16
```

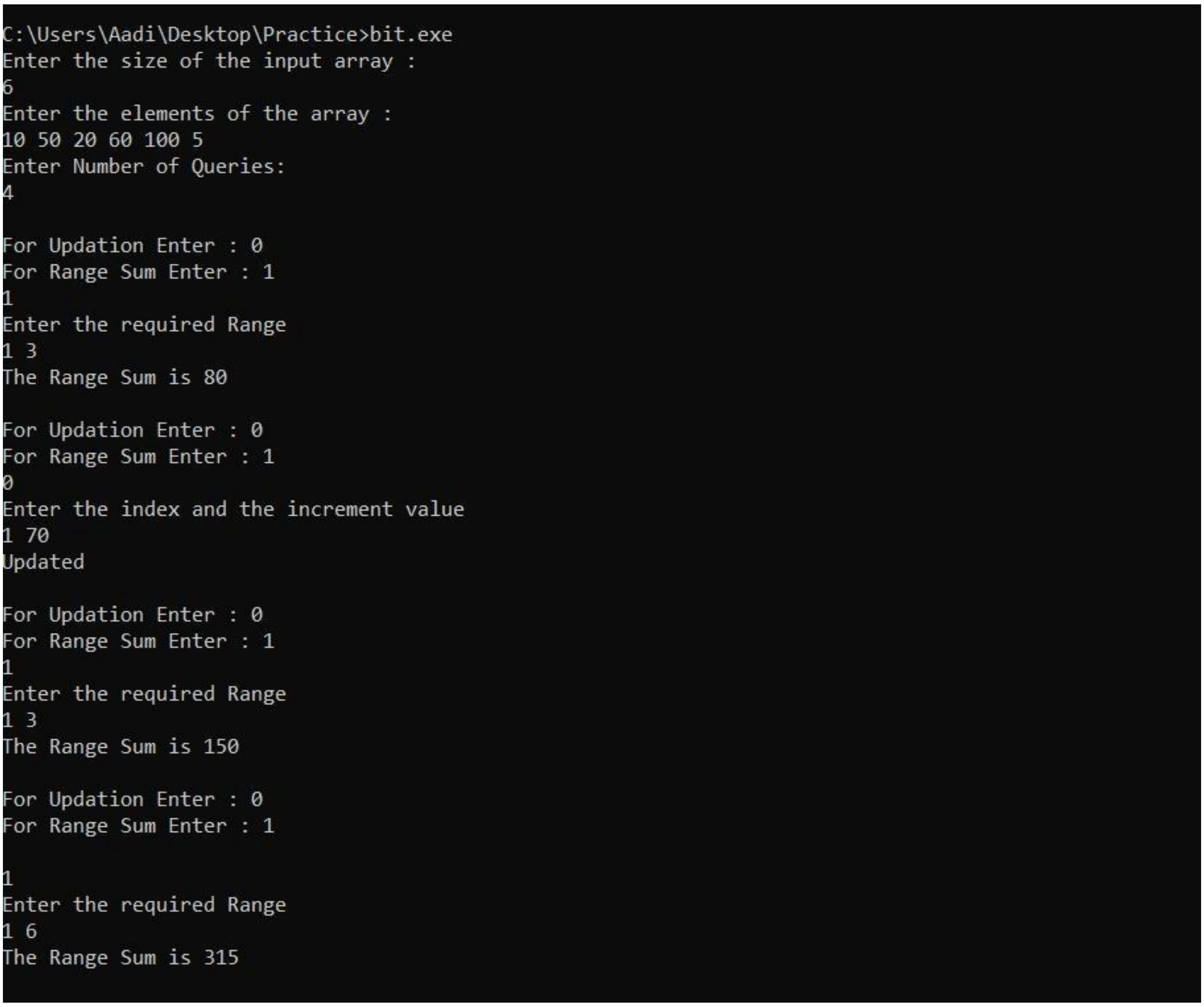


```
PS D:\Aruna\C++ test1> python -u "d:\Aruna\C++ test1\BIT.py"
Array: [3, 2, -1, 6, 5, 4, -3, 3, 7, 2, 3]
BITarr:  [0, 3, 5, -1, 10, 5, 9, -3, 19, 7, 9, 3]
Prefix sum of elements in range [0,10]: 31
Prefix sum of elements in range [0,6]: 16
Range sum of elements in [2,5]: 14

Change the value of element at pos 4 to 7
Updated Array: [3, 2, -1, 6, 7, 4, -3, 3, 7, 2, 3]

Prefix sum of elements in range [0,10]: 33
Prefix sum of elements in range [0,6]: 18
Range sum of elements in [2,5]: 16
```

C++ implementation of BIT data structure:

```
//Binary Indexed Tree:
#include<bits/stdc++.h>
using namespace std;

int getSum(int *bit,int n,int k){
    int ans = 0;
    for(int i=k;i>0; i-=(i & -i)){
        ans+=bit[i];
    }
    return ans;
}

void update(int *bit,int n,int index,int val){
    for(int i= index;i<=n;i += (i & -i)){
        bit[i]+= val;
    }
}

int main(){
    // ios:: sync_with_stdio(false);
 //     cin.tie(0);

    cout<<"Enter the size of the input array : "<<endl;
    int n;
    cin>>n;
    //init block
    int bit[n+1];
    memset(bit,0,sizeof(bit));

    cout<<"Enter the elements of the array : "<<endl;
    int * data = new int [n+1];
    for(int i=1;i<=n;++i){
        int e;
        cin>>e;
        data[i] = e;
        update(bit,n,i,e);
    }
    //queries
    cout<<"Enter Number of Queries: "<<endl;
    int q;
    cin>>q;

        while(q--){
          cout<<"For Updation Enter : 0\nFor Range Sum Enter : 1 "<<endl;
        int option;
        cin>>option;
        if(option){
        cout<<"Enter the required Range "<<endl;
        int l,r;
        cin>>l>>r;
        cout<<"The Range Sum is "<<getSum(bit,n,r) - getSum(bit,n,l-1)<<endl;
        }
        else{
            int index,val;
        cout<<"Enter the index and the increment value "<<endl;
            cin>>index>>val;
            update(bit,n,index,val);
            cout<<"Updated"<<endl;
        }
  }
}
```

OUTPUT:



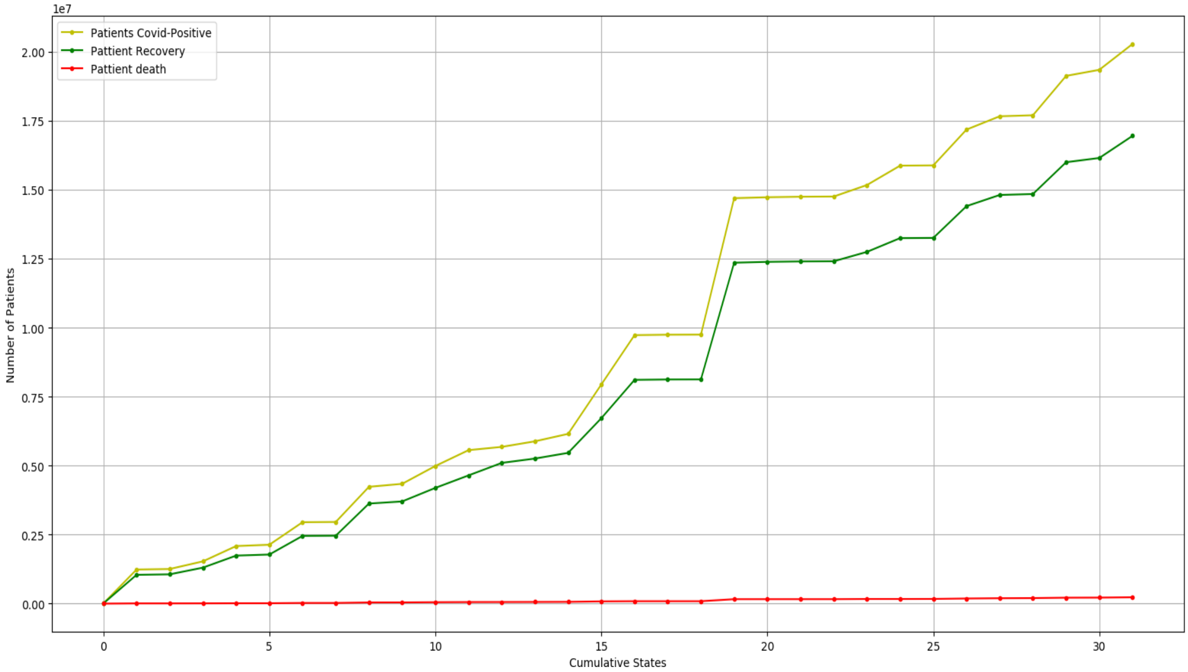
```
C:\Users\Aadi\Desktop\Practice>bit.exe
Enter the size of the input array :
6
Enter the elements of the array :
10 50 20 60 100 5
Enter Number of Queries:
4

For Updation Enter : 0
For Range Sum Enter : 1
1
Enter the required Range
1 3
The Range Sum is 80

For Updation Enter : 0
For Range Sum Enter : 1
0
Enter the index and the increment value
1 70
Updated

For Updation Enter : 0
For Range Sum Enter : 1
1
Enter the required Range
1 3
The Range Sum is 150

For Updation Enter : 0
For Range Sum Enter : 1

1
Enter the required Range
1 6
The Range Sum is 315
```

Application of BIT:

```python
# -*- coding: utf-8 -*-

#import and collecting data-Set:
#https://data.gov.in/major-indicator/covid-19-india-data-source-mohfw
import xlrd
book = xlrd.open_workbook('C:/Users/Aadi/Desktop/Sample.xlsx')
sheet = book.sheet_by_name('Sheet1')
positive = [sheet.cell_value(r, 0) for r in range(sheet.nrows)]
recovery = [sheet.cell_value(r, 1) for r in range(sheet.nrows)]
death = [sheet.cell_value(r, 2) for r in range(sheet.nrows)]


#Binary Index Tree
```

```
def getsum(BITTree,i):
    s = 0
    i = i+1
    while i > 0:
        s += BITTree[i]
        i -= i & (-i)
    return s

def updatebit(BITTree , n , i ,v):
    i += 1
    while i <= n:
        BITTree[i] += v
        i += i & (-i)

def construct(arr, n):
    BITTree = [0]*(n+1)

    for i in range(n):
        updatebit(BITTree, n, i, arr[i])
    return BITTree

BITTree_positive = construct(positive,len(positive));
BITTree_recovery = construct(recovery,len(recovery));
BITTree_death = construct(death,len(death));
#construct cumulative sum:

c_positive = [];
c_recovery =[];
c_death =[];
for i in range(len(positive)):
    c_positive.append(getsum(BITTree_positive,i));
    c_recovery.append(getsum(BITTree_recovery,i));
    c_death.append(getsum(BITTree_death,i));


#plot cumulative graph:

import matplotlib.pyplot as plt
plt.plot(c_positive ,color = 'y', marker = '.' , label = "Patients Covid-Positive");
plt.plot(c_recovery , color = 'g', marker = '.',label = 'Pattient Recovery');
plt.plot(c_death , color = 'r', marker = '.',label = 'Pattient death');

plt.xlabel("Cumulative States")
plt.ylabel("Number of Patients")
plt.legend();
plt.grid();
plt.show();
```

OUTPUT GRAPH:


1e7
Patients Covid-Positive
Pattient Recovery
Pattient death
Number of Patients
Cumulative States