12/19/2022

Binary Indexed Tree

Quiz 4 Equivalent

Name: K.M. TAHLIL MAHFUZ FARUK

Student ID: 200042158

Course: CSE 4303(Data Structure)

K.M. Tahlil Mahfuz Faruk
ISLAMIC UNIVERSITY OF TECHNOLOGY

Binary Indexed Tree:

A Binary Indexed Tree (aka. Fenwick Tree) is a data structure that helps to get the sum of a particular range more efficiently by setting values in a vector/array using prefix sum.

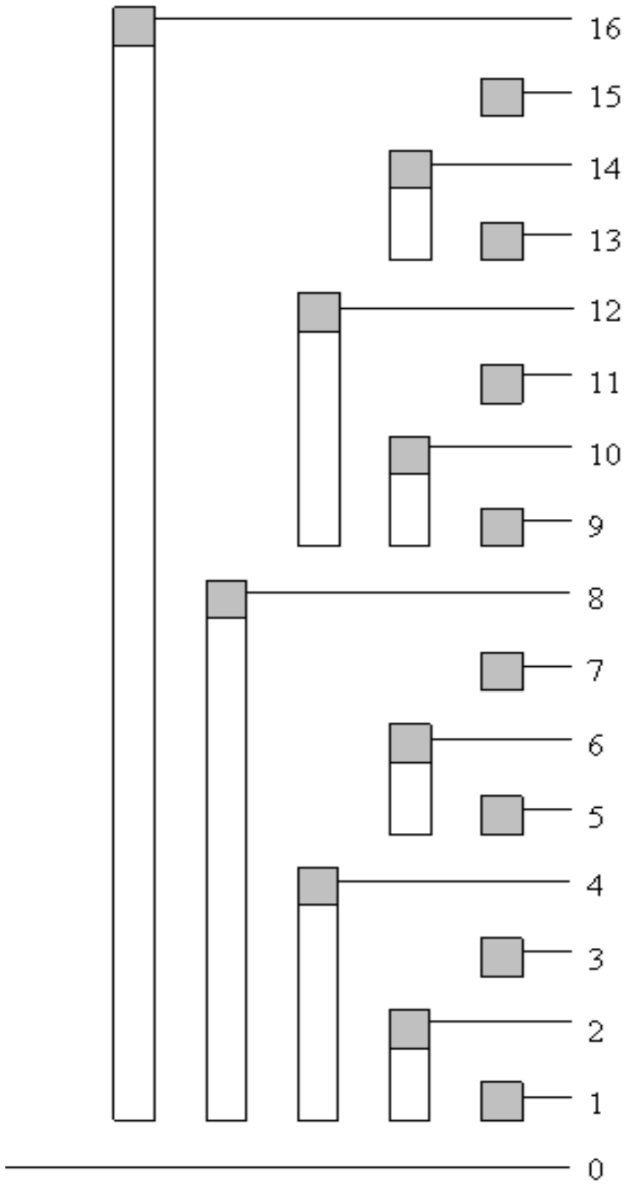

Image 1.3 – tree of responsibility for indices (bar shows range of frequencies accumulated in top element)

The figure represents the responsibility of particular ranges. For example, for range 16 the responsible ranges would be 16, 15, (14-13), (12-9) & (8-1). If we sum up these ranges then we will get the sum of 1 to 16 range numbers.

Again, if we take range 1 to 3, then the responsible ranges would be 3 & (2-1).

This procedure will satisfy all the ranges that can be occupied.

BIT Standard Operations:

A binary indexed tree can have the following standard operations.

- Insert
- Update
- RSQ(Range Sum Query)
- GetSum
- Print

Advantages:

- BIT is easier to implement than Segment tree.
- The concept of bitwise operation is used that is really efficient than any other approach.
- BIT needs linear memory space O(N).
- Total time complexity including construction would be O(nlog(n)).
- It uses prefix sum more efficiently.

Disadvantages:

- Jumping between the ranges is hard to understand (index+=index & -index).

Sample code of a basic implementation of BIT:

```
#include<bits/stdc++.h>
using namespace std;


class BIT{
    vector<int>v,tree;
    public :
    BIT(int size){
        v.resize(size);
        tree.resize(size);
    }

    void insert(vector<int>vec){
        v=vec;
        for(int i=1;i<=v.size();i++){
            update(i,v[i]);
        }
    }

    void update(int index,int val){
        while(index<=v.size()){
            tree[index]+=val;
            index+=(index & (-index));
        }
    }

    int rsq(int from,int to){
        return getSum(to)-getSum(from-1);
    }

    int getSum(int index){
        int sum=0;
        while(index>0){
            sum+=tree[index];
            index-=index & -index;
        }
        return sum;
    }

    void print(){
        for(auto e:tree){
            cout<<e<<" ";
        }
```

```
        cout<<endl;
    }

};


signed main()
{
    int n;cin>>n;
    vector<int>v(n+1);
    for(int i=1;i<=n;i++){cin>>v[i];}
    BIT bittree= BIT(v.size());
    bittree.insert(v);
    bittree.print();

    cout<<bittree.getSum(10)<<endl;  //19
    cout<<bittree.rsq(1,3);          //3

    return 0;
}

// input:
// 16
// 1 0 2 1 1 3 0 4 2 5 2 2 3 1 0 2
```

Complexity Analysis:

Construction	O(n)
Update	O(log(n))
GetSum	O(log(n))
RSQ (Range Sum Query)	O(log(n))
Print Tree	O(n)

Space Complexity

Space Complexity	O(N)

Time complexity will differ in terms of best cases and worst cases.

Best Case Complexity: O(logN)

Worst Case Complexity: O(N)

Use cases:

Given a set of numbers. We need to find the sum of the set of number for a particular range.

Constraints:

- *Memory limit: 20MB (Very small)*
- *Time limit: 1.5s(Very small)*
- *1<n<20e10, n= Total size of numbers*
- *1<m<10e4, m= Number of queries*

This problem can be solved using segment tree. But it will fail because of the memory limit. In this case we can only use BIT(Binary Indexed Tree) because it's memory complexity is linear O(n).