Binary Indexed Tree

Chenqi Wang

09/01/2022

1 Definitions

On array $a[1, ..., n]$, we can build the corresponding Binary Indexed Tree $c[1, ..., n]$:

$$c[i] := \sum_{j=i-lowbit(i)+1}^{i} a[j]$$

where $lowbit(i) = i \ \& \ (-i)$.

$c[i + lowbit(i)]$ is the parent of $c[i]$. This fact is useful for initialization:

$$c[j] = a[j] + \sum_{i+lowbit(i)==j} c[i]$$

2 Problems & Solutions

Problem: https://leetcode.com/problems/range-sum-query-mutable/

```
class NumArray {
    vector<int> c_; // Binary Indexed Tree
    int n_;
    vector<int> nums_;
public:
    inline int lowbit(int x) {return x & (-x);}

    NumArray(vector<int>& nums) {
        nums_ = nums;
        n_ = nums.size();
        c_ = vector<int>(n_ + 1);

        for (int i = 1; i <= n_; ++i) {
            c_[i] += nums[i - 1];
```

```
15
16              int j = i + lowbit(i);
17              if (j <= n_) c_[j] += c_[i];
18          }
19      }
20
21      void update(int index, int val) {
22          int add = val - nums_[index];
23          nums_[index] = val;
24
25          int x = index + 1;
26          while (x <= n_) {
27              c_[x] += add;
28              x += lowbit(x);
29          }
30      }
31
32      inline int prefixSum(int x) {
33          int ret = 0;
34          while (x >= 1) {
35              ret += c_[x];
36              x -= lowbit(x);
37          }
38
39          return ret;
40      }
41
42      int sumRange(int left, int right) {
43          return prefixSum(right + 1) - prefixSum(left);
44      }
45  };
46
47  /**
48   * Your NumArray object will be instantiated and called as such:
49   * NumArray* obj = new NumArray(nums);
50   * obj->update(index,val);
51   * int param_2 = obj->sumRange(left,right);
52   */
```

source: https://oi-wiki.org/ds/fenwick/