Binary indexed tree (Fenwick tree)

2015年12月6日 (https://www.hrwhisper.me/binary-indexed-tree-fenwick-tree/) hrwhisper (https://www.hrwhisper.me/author/hrsay/) 2 Comments (https://www.hrwhisper.me/binary-indexed-tree-fenw tree/#comments) 3,241 views

本文简单介绍Binary indexed tree (Fenwick tree)

Fenwick tree

它又叫 **Binary indexed tree** , 也叫树状数组。

能在log(n)查询区间和，并且在log(n)时间内进行结点更新操作。

lowbit(x)函数

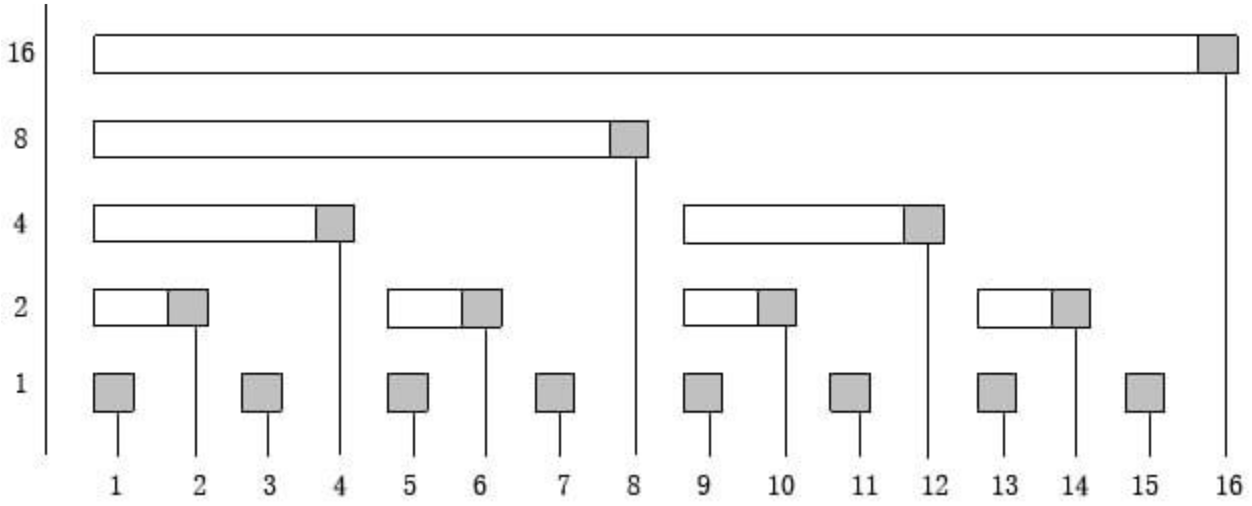
定义lowbit(x)为x的二进制表达式中最右边的1所对应的值。

比如，1234的二进制是0100 1101 0010 lowbit(1234)=2，在程序的实现中，

Lowbit(x)=x&-x;（为什么这样写呢？因为计算机内部采用补码表示，-x是x按位取反，
数+1的结果）

树的结构图

让我们来看看图：横坐标是x, 纵坐标是lowbit(x)



(https://www.hrwhisper.me/wp-content/uploads/2015/11/fenwick_tree_binary_index_tree.jpg)

对于节点x，

- 为左子结点，则父结点的编号是x+lowbit(x)，
- 为右子结点，则父结点的编号是x-lowbit(x)

设C[i] 为以i结尾的水平长条内的元素之和，如c[6]=a5+a6。

- 顺着结点I往左走，边走边往上爬，沿途经过的c[i]所对应的长条不重复不遗漏的
所有需要累加的元素。
    - 如sum(6) = c[6] + c[4]
- 如果修改了一个a[i] ,那么从c[i]往右走，边走边网上爬，沿途修改所有结点对应的
可。
    - 如a[1] + 1 那么 c[1] + 1, c[2]+1,c[4]+1………一直到最大值。

用C++ 的代码如下：

```
inline int lowbit(int x) { return x&(-x) ; }

int sum(int x)
{
    int ans=0;
    while(x>0)
    {
        ans+=C[x];
       x-=lowbit(x);
    }
    return ans;
}

void add(int x,int d)
{
    while(x<=N)
    {
        C[x]+=d;
       x+=lowbit(x);
    }
}
```

实现代码

写成类的话：

C++

```
class FenwickTree {
    vector<int> sum_array;
    int n;
    inline int lowbit(int x) {
        return x & -x;
    }

public:
    FenwickTree(int n) :n(n), sum_array(n + 1, 0) {}

    void add(int x, int val) {
        while (x <= n) {
            sum_array[x] += val;
            x += lowbit(x);
        }
    }

    int sum(int x) {
        int res = 0;
        while (x > 0) {
            res += sum_array[x];
            x -= lowbit(x);
        }
        return res;
    }
};
```

Python

```python
class FenwickTree(object):
    def __init__(self, n):
        self.sum_array = [0] * (n + 1)
        self.n = n

    def lowbit(self, x):
        return x & -x

    def add(self, x, val):
        while x <= self.n:
            self.sum_array[x] += val
            x += self.lowbit(x)

    def sum(self, x):
        res = 0
        while x > 0:
            res += self.sum_array[x]
            x -= self.lowbit(x)
        return res
```

> *本博客若无特殊说明则由 hrwhisper (https://www.hrwhisper.me) 原创发布*
>
> *转载请点名出处：细语呢喃 (https://www.hrwhisper.me) > Binary indexed tree (Fenwick tree) (https://www.hrwhisper.me/binary-indexed-tree-fenwick-tree/)*
>
> *本文地址：https://www.hrwhisper.me/binary-indexed-tree-fenwick-tree/*

> *(https://www.hrwhisper.me/binary-indexed-tree-fenwick-tree/)*

分享到： 新浪微博 微信 QQ空间 人人 腾讯微博 豆瓣 印象笔记

算法笔记 (https://www.hrwhisper.me/category/study/algorithm-notes/) Data Structure (https://www.hrwhis /tag/data-structure/), 算法 (https://www.hrwhisper.me/tag/algorithm/). permalink (https://www.hrwhisper.me/bi indexed-tree-fenwick-tree/).

‹ leetcode Count of Smaller Numbers After Self (https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/)

leetcode Remove Duplicate Letters › (https://www.hrwhisper.me/leetcode-remove-duplicate-le

2 thoughts on “Binary indexed tree (Fenwick tree)”

Pingback: leetcode Count of Range Sum - 细语呢喃 (https://www.hrwhisper.me/leetcode-count-of-range

Pingback: Binary indexed tree (Fenwick tree) – binadog (http://ec2-54-213-80-199.us-west-2.compute.amazonaws.com/blog/2016/01/07/binary-index-tree/)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

(http://weibo.com/murmured)

(https://twitter.com/hrwhisper)

(https://instagram.com/hr_say/)

(https://github.com/hrwhisper)

(https://www.hrwhisper.me/feed)

Csdn (http://blog.csdn.net/murmured) 博客园 (http://www.cnblogs.com/murmured/)

Lofter (http://hrsay.lofter.com/) 知 乎 (http://www.zhihu.com/people/hrwhisper)

豆瓣 (http://www.douban.com/people/hrwhisper/)

努力的人本身就有奇迹 | 快乐是我们共同的

by hrwhipse