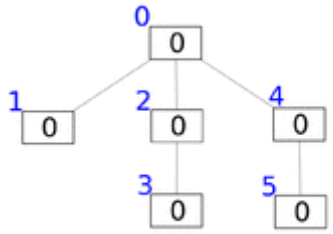
WIKIPEDIA

树状数组

维基百科，自由的百科全书

树状数组或**二叉索引树**（英語：Binary Indexed Tree），又以其发明者命名为Fenwick树，最早由Peter M. Fenwick于1994年以A New Data Structure for Cumulative Frequency Tables[1]为题发表在SOFTWARE PRACTICE AND EXPERIENCE。其初衷是解决数据压缩里的累积频率（Cumulative Frequency）的计算问题，现多用于高效计算数列的前缀和，区间和。它可以以$O(\log n)$的时间得到任意前缀和$\sum_{i=1}^{j} A[i], 1 <= j <= N$，并同时支持在$O(\log n)$时间内支持动态单点值的修改。空间复杂度$O(n)$。



根据数组[1, 2, 3, 4, 5]来创建对应的树状数组

目录

结构起源

按照Peter M. Fenwick的说法，正如所有的整数都可以表示成2的幂和，我们也可以把一串序列表示成一系列子序列的和。采用这个想法，我们可将一个前缀和划分成多个子序列的和，而划分的方法与数的2的幂和具有极其相似的方式。一方面，子序列的个数是其二进制表示中1的个数，另一方面，子序列代表的f[i]的个数也是2的幂。[2][3][4]

基本操作

预备函数

定义一个Lowbit函数，返回参数转为二进制后,最后一个1的位置所代表的数值.

例如,Lowbit(34)的返回值将是2；而Lowbit(12)返回4；Lowbit(8)返回8。

将34转为二进制,为0010 0010,这里的"最后一个1"指的是从2^0位往前数,见到的第一个1,也就是2^1位上的1.

程序上，((Not I)+1) And I表明了最后一位1的值,

仍然以34为例,Not 0010 0010的结果是 1101 1101(221),加一后为 1101 1110(222), 把 0010 0010与1101 1110作AND,得0000 0010(2).

Lowbit的一个简便求法:（C++）

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

新建

定义一个数组 BIT，用以维护A的前缀和，则:

$$BIT_i = \sum_{j=i-lowbit(i)+1}^{i} A_j$$

具体能用以下方式实现：（C++）

```
void build()
{
    for (int i = 1; i <= MAX_N; i++)
    {
        BIT[i] = A[i - 1];
        for (int j = i - 2; j >= i - lowbit(i); j--)
            BIT[i] += A[j];
    }
}
//注：这里的求和将汇集到非终端结点（D00形式）
//BIT中仅非终端结点i值是 第0~i元素的和
//终端结点位置的元素和，将在求和函数中求得
//BIT中的index，比原数组中大1
```

修改

假设现在要将$A[i]$的值增加delta，

那么，需要将$BIT[i]$覆盖的区间包含$A[i]$的值都加上delta，

这个过程可以写成递归，或者普通的循环。

需要计算的次数与数据规模N的二进制位数有关，即这部分的时间复杂度是O(LogN)

修改函数的C++写法

```
void edit(int i, int delta)
{
    for (int j = i; j <= MAX_N; j += lowbit(j))
        BIT[j] += delta;
}
```

求和

假设我们需要计算$\sum_{i=1}^{k} A_i$的值。

1. 首先,将ans初始化为0，将i初始化为k

2. 将ans的值加上BIT[i]
3. 将i的值减去lowbit(i)
4. 重复步骤2～3，直到i的值变为0

求和函数的C/C++写法

```
int sum (int k)
{
    int ans = 0;
    for (int i = k; i > 0; i -= lowbit(i))
        ans += BIT[i];
    return ans;
}
```

复杂度

初始化复杂度最优为$O(N)$

单次询问复杂度$O(\log N)$，其中N为数组大小

单次修改复杂度$O(\log N)$，其中N为数组大小

空间复杂度$O(N)$

应用

求逆序数[5]

逆序数是一个数列中在它前面有比它大的个数。如4312的逆序数是0+1+2+2=5。

可以先把数列中的数按大小顺序转化成1到n的整数，使得原数列成为一个$1, 2, \ldots, n$的排列P，创建一个树状数组，用来记录这样一个数组A（下标从1算起）的前缀和：若排列中的数i当前已经出现，则$A[i]$的值为1，否则为0。初始时数组A的值均为0，从排列中的最後一个数开始遍历，每次在树状数组中查询有多少个数小于当前的数$P[j]$（即用树状数组查询数组A目前$P[j]-1$个数的前缀和）并加入计数器，之后对树状数组执行修改数组A第$P[j]$个数值加1的操作。

参考文献

1. Peter M. Fenwick. A new data structure for cumulative frequency tables. Software: Practice and Experience. 1994, **24** (3): 327–336. doi:10.1002/spe.4380240306.
2. Binary indexed tree-树状数组 (http://duanple.blog.163.com/blog/static/709717672008113111314583 2/)
3. Binary Indexed Trees (http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=binaryIndexedTrees)
4. TopCoder树状数组教程的译文 (http://hawstein.com/posts/binary-indexed-trees.html)
5. http://blog.csdn.net/cattycat/article/details/5640838