树状数组

树状数组

树状数组或二叉索引树（BIT, Binary Indexed Tree），又叫Fenwick树，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。

树状数组是一种利用**数的二进制特征**进行检索的树状结构，是查询、修改操作的时间复杂度均为O(logn)的数据结构。

树状数组

问题模型：

对数组a[1，....，n]，需维护以下2种操作：

1.查询：询问前i个元素的累加和

2.修改：更新a[i]的值，1 ≤ i ≤ n

朴素的方法：

方法1：查询操作，需遍历 1到 i 的元素并计算出累加和，时间复杂度为O(n)；直接更新操作 a[i] = x，时间复杂度为O(1) ；

方法2：创建一个大小为 n 的新数组，并且在新数组的第 i 个位置保存前 i 个元素的累加和，查找操作在O(1)的时间内完成；更新操作的时间复杂度为O(n) 。

树状数组

问题：若修改、查询操作次数都比较多，可否有更高效的方法?

树状数组是一个查询和修改操作时间复杂度都为O(log*n*)的数据结构，主要用于数组的单点修改和区间求和。

树状数组

对于长度为 **n** 的数列 A ={a_1, a_2, ..., a_n} ,

- 前缀和 sum(k) = a [1] + a[2] + . . . + a[k] , $k \le n$

- 区间和查询 rangeSum(i, j) :

  a [i] + . . . + a[j] = sum(j) - sum(i - 1) $1 \le i, j \le n$

树状数组

■ 基本思想：

➢ 所有的整数都可以表示成2的幂的和（用二进制表示十进制数），Peter M. Fenwick 受此启发，把一串序列的和表示成一系列子集（或子序列）的和。

例如，$(7)_{10}=2^2+2^1+2^0$，那么求前缀和sum[7]可以分解为3个子集的和，如何分解？

子集标号 i	1	2	3	4	5	6	7	8
包含的数组元素	a[1]	a[1] a[2]	a[3]	a[1] a[2] a[3] a[4]	a[5]	a[5] a[6]	a[7]	a[1] a[2] … a[8]

子集划分

树状数组

前缀和与子集和

下标 i	1	2	3	4	5	6	7	8
a[i]	1	2	3	4	5	6	7	8
前缀和 sum[i]	1	3	6	10	15	21	**28**	36
子集	a[1]	a[1] a[2]	a[3]	a[1] a[2] a[3] a[4]	a[5]	a[5] a[6]	a[7]	a[1] a[2] … a[8]
子集和数组C[i]	1	3	3	10	5	11	7	36

那么，求前缀和sum[7]，可以分解为求C[7]、 C[6]、 C[4]这3个子集的和；求前缀和sum[[8]的和只需求C[8]的值。

树状数组

- 将一个前缀和划分成多个子集的和，而划分的方法与数的2的幂和具有极其相似的方式。

例：$(7)_{10}= 2^2+2^1+2^0$

$(7)_{10}=(111)_2$

3个子集

※发现：

（1）子集的个数是其下标二进制表示中1的个数。

（2）子集中含有的原数组中的元素个数也是2的幂。

树状数组

- 将一个前缀和划分成多个子集的和，而划分的方法与数的2的幂和具有极其相似的方式。

假设整数n，其二进制表示为

$n = 2^{i1}+2^{i2}+...+2^{im}$，其中i1，i2，...，im代表二进制表示下位为1的索引下标值，

并且i1>i2>...>im>=0。那么，可以将区间[1，n]划分成m个子集：

$[1, 2^{i1}]$，$[2^{i1}+1, 2^{i1}+2^{i2}]$，....，$[2^{i1}+2^{i2}+...+2^{im-1}+1, n]$

例如：

$n=7=(111)_2= 2^2+2^1+2^0$，那么 区间[1，7] 可以划分成 [1，4]、 [5，6] 和 [7，7] 这3个子集。

树状数组

子集以及其所包含的A数组元素个数之间的关系如下表所示：

下标 i	1	2	3	4	5	6	7	8
i 的二进制表示	1	10	11	100	101	110	111	1000
子集中元素个数	1	2	1	4	1	2	1	8
元素个数的二进制表示	1	10	1	100	1	10	1	1000

※发现：元素个数的二进制表示就是下标i的二进制表示中最低位的1的位权。

这里，最低位的1的位权可以利用**低位技术（Lowbit）**求得。

树状数组

一个重要技术—低位技术（Lowbit）：

lowbit(index)函数的功能就是求数index的二进制表示中最低位1的位权值。

例如：

若 index = 6，则 lowbit(6) = 2

若 index = 7，则 lowbit(7) = 1

树状数组

lowbit(index)函数可以借助位运算得以实现：

方法1：

$$C(index) = index \,\&\, (-index\,)$$

方法2：

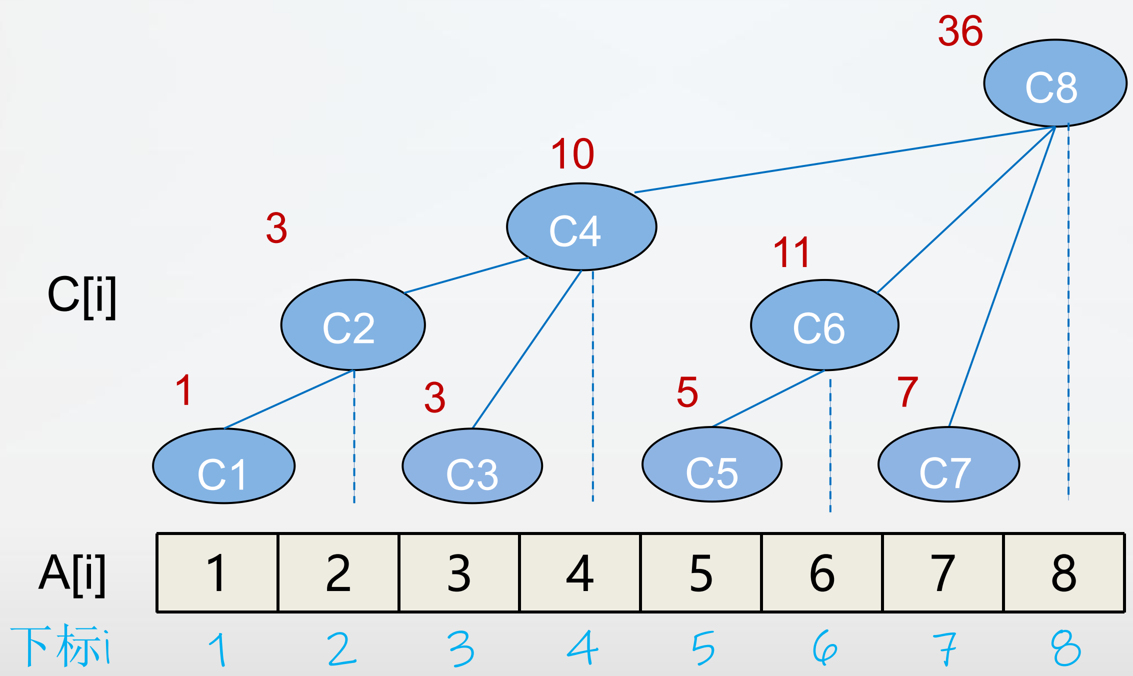
$$C(index) = x - (\,index \,\&\, (index-1))$$

例如： $x = (6)_{10} = (110)_2$，则 $lowbit(110) = (10)_2 = (2)_{10}$

树状数组

定义： 原数组A ={a_1, a_2, ..., a_n}， 定义**C[n]**为其树状数组（BIT, Binary Indexed Tree），

其中C [i]表示A[i-lowbit(i)+1]至A[i]的和 （其中，1<=i<=n）。



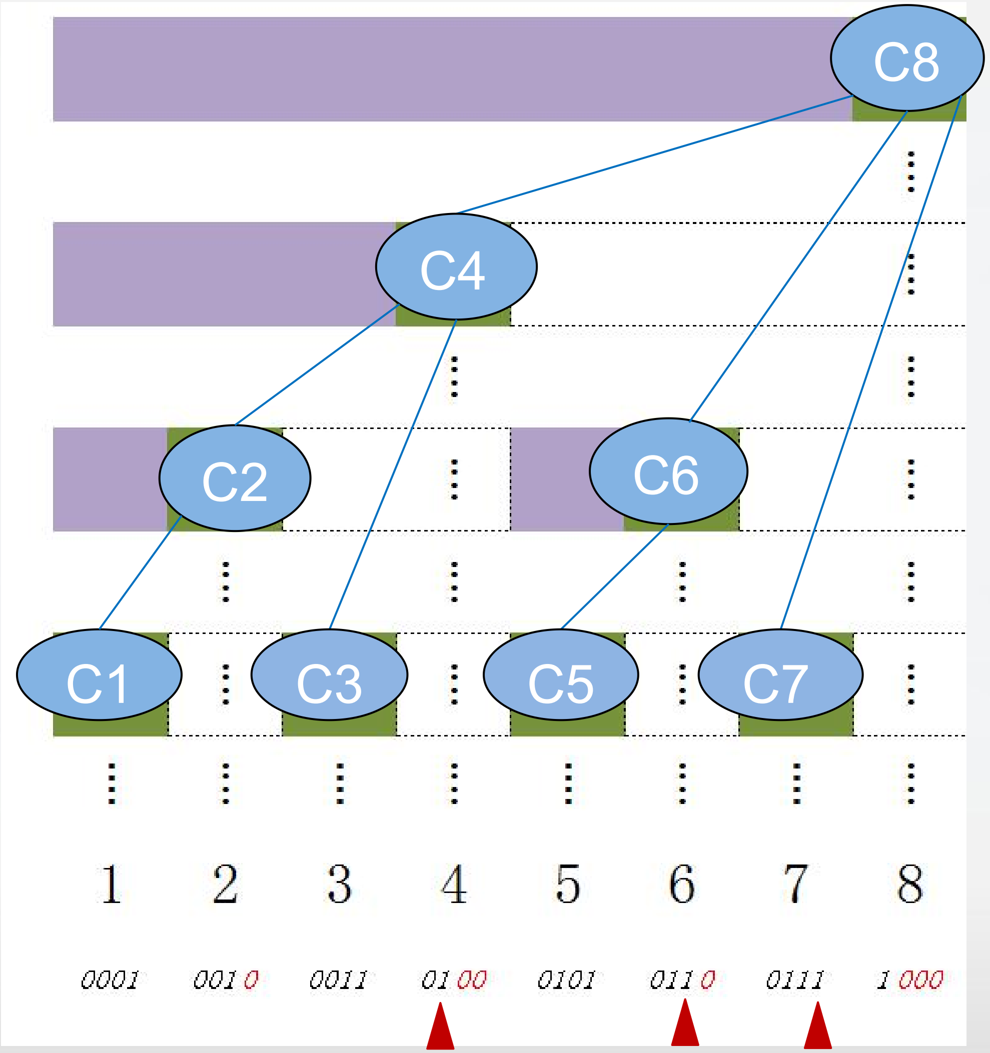
树状数组

例：求前缀和，

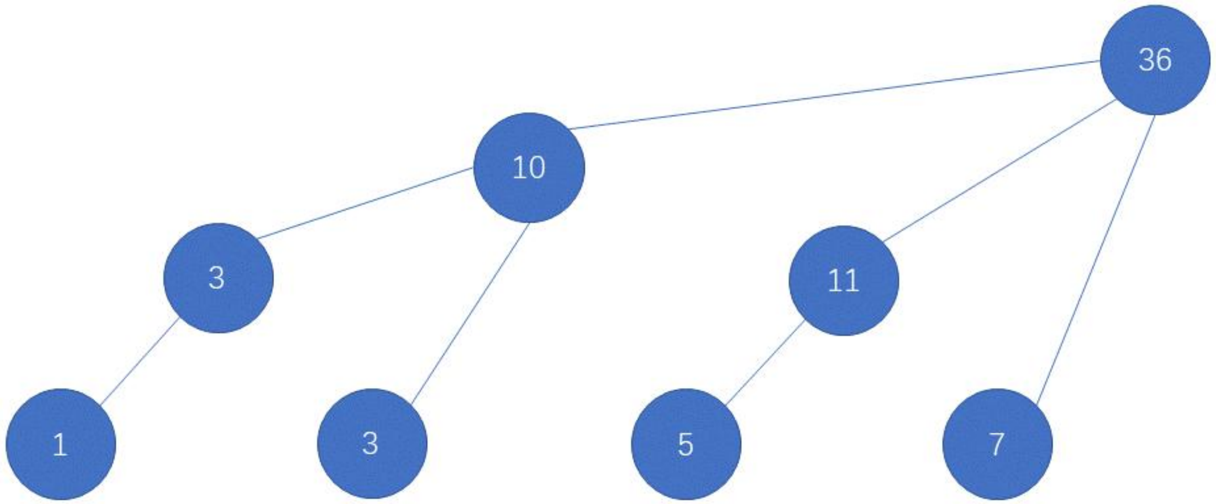
sum(4)=C[4]

sum(6)=C[4]+C[6]

sum(7)=C[4]+C[6]+C[7]



树状数组



		1	2	3	4	5	6	7	8
	Index	1	2	3	4	5	6	7	8
A[index]	Original Value	1	2	3	4	5	6	7	8
C[index]	BIT Value	1	3	3	10	5	11	7	36
index	Binary bit	0001	0010	0011	0100	0101	0110	0111	1000
	Low bit	0001	0010	0001	0100	0001	0010	0001	1000

树状数组

- 基本操作
  - 查询前缀和 Query
  - 修改元素值 Change
  - 创建

树状数组

- 查询前缀和

求a[1]+a[2]+...+a[i]的值

```
int Query ( int index)
{
    ans=0;
    While ( index > 0 ) {
        ans  +=   C[index];
        index -=  lowbit(index);
    }
    return ans;
}
```

说明： 需要相加的子集个数是index的二进制表示中“1”的个数，所以查询操作的时间复杂度是O(log*n*)。

树状数组

相关问题1： 查询任意区间和A[x....y]？

$$A[x....y] = A[x] + A[x+1] + ... + A[y]$$

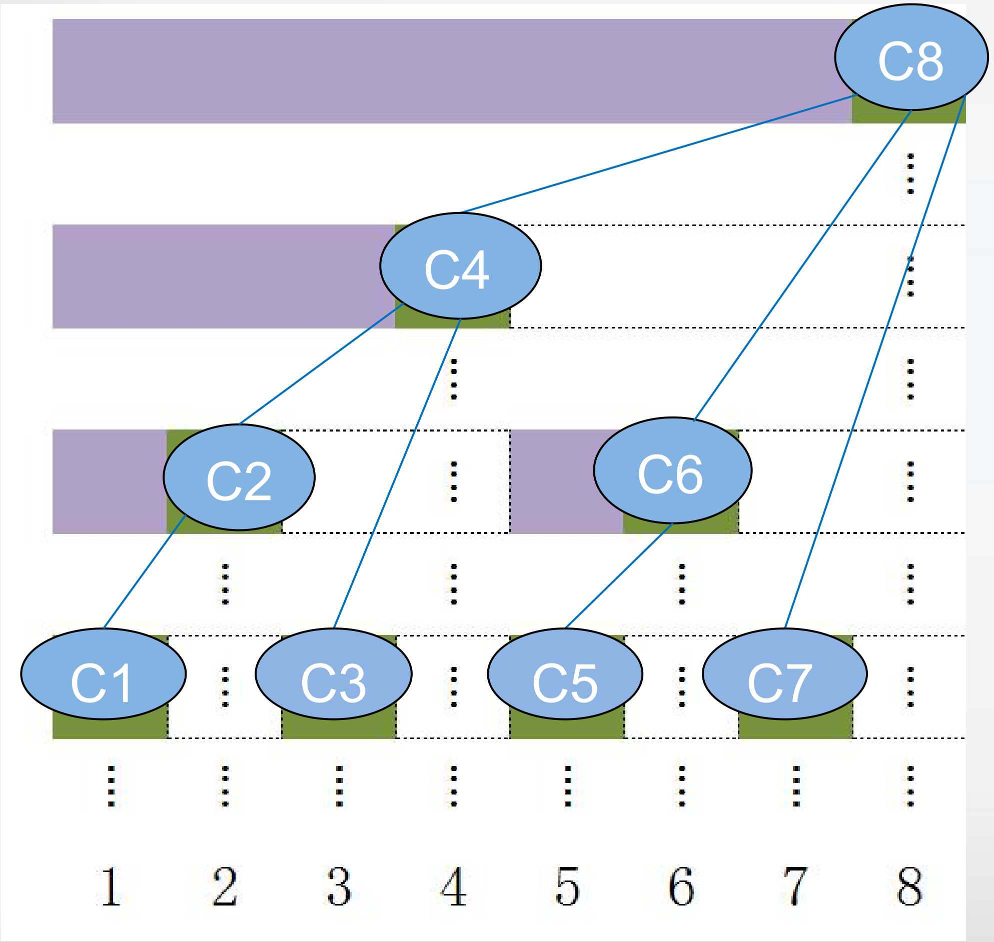
$$= Query(y) - Query(x-1)$$

相关问题2： 利用C数组求原数组A的某个元素值A[i]=？

$$A[i] = Query(i) - Query(i-1)$$

树状数组

- 修改元素值



注意： 需维护的“修改”是针对数组A的单个元素A[i]，所以要考察哪些子集包含A[i]。

例如，修改A[1]，则分别需要修改C[1]、C[2]、C[4]、C[8]；修改A[3]，则分别需要修改C[3]、C[4]、C[8]。

规律：“从子结点到父结点”，即从结点C[index]开始，向树根的方向寻找其系列祖先结点，修改这些结点的值：

parent[index] = index +lowbit[index]

树状数组

- 修改元素值

```
void change( int index, int delta)
{
    While ( index <= n ) {
        C[index] += delta;
        index += lowbit(index);
    }
}
```

说明： 由于树的深度至多是$\log n+1$，所以修改操作的时间复杂度是$O(\log n)$。

树状数组

- 创建

方法1：利用修改元素值的操作

方法2：

借助一个前缀和数组 pre[index]= A[1]+A[2]+…+A[x]

C[index] = pre[index] – pre[index - lowbit(index)]

方法3：

C[index] = A[index - lowbit(index)+1] + …..+ A[index]

树状数组

- 树状数组扩展到高维的情形

二维数组a[1...n, 1...n]，维护以下两种操作：

（1）修改更新：给a[i, j] 加上一个增量；

（2）查询：询问矩形区域a[1...x, 1...y] 的和，即 $\sum_{i=1}^{x}\sum_{j=1}^{y}a[i,j]$

同样，用一个二维数组C[n][n]维护被分割的“子集”之和，可以模仿一维情形下的定义，将C[i][j]定义为 $C[x,y]=\sum_{i=x-lowbit(x)+1}^{x}\sum_{j=y-lowbit(y)+1}^{y}a[i,j]$

树状数组

◆ **特点：**

1. 树状数组利用了区间和的“可加性”，可以较高效率地处理区间和的查询。
2. 树状数组在查询前缀和以及修改操作的时间复杂度均是O(log*n*)。
3. 更突出的特点是其编程的简洁性，使用lowbit技术可以在很短的几步操作中完成树状数组的核心操作，其代码效率较高。