LOGO

树状数组

—— 高级数据结构之二

《数据结构与算法实习》

北京大学信息学院
张路

2017.10

LOGO

先考虑这样一个问题

- 给定一个n个元素的数组a[1],a[2]...a[n]
- 定义前缀和S[K] = a[1] + a[2] + ... + a[K]
- 一共q个询问，K1,K2...Kq，求S[K1],S[K2]...S[Kq]

LOGO

前缀和问题

- Brute-Force?
- 预处理
- 预处理O(n)   询问O(q)

LOGO

前缀和问题

- 如果a数组可以修改？（修改和询问一共q次）
- Brute-Force O(nq)

LOGO

树状数组

- 预处理O(n)
- 修改、询问O(q lgn)

LOGO

例题

- 给定一个初始值都为0长度为n的序列,有如下操作:
- 1)修改操作以'C'跟两个整数的形式给出,次数为c
  - 如C 2 3表示将第2个数的值加3;
- 2)询问操作以'Q'跟两个整数的形式给出,次数为q
  - 如Q 1 4表示求第1个数到第4个数的区间和;

- 输入
- 6 //数组长度
- C 2 4
- Q 1 4
- C 3 -2
- C 5 10
- C 2 -1
- Q 2 6

输出
4
11

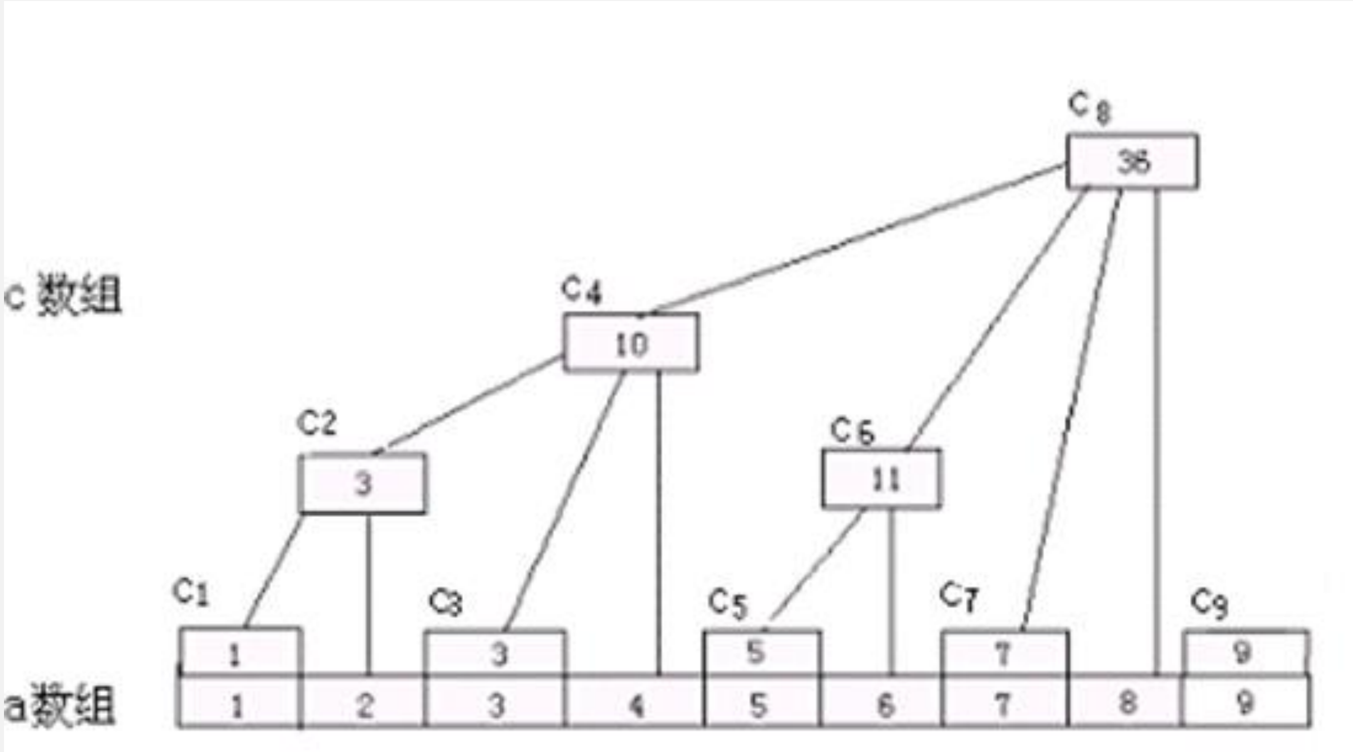
LOGO

树状数组

- 设原数组为a[1],a[2]...a[n]
- 树状数组为c[1],c[2]...c[n]
- 用c数组求a数组的部分和

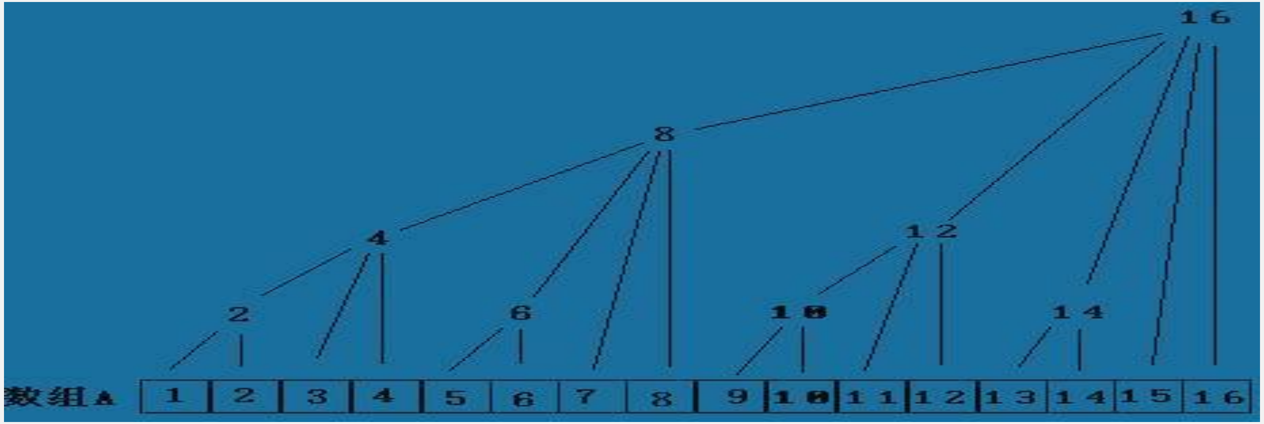
LOGO

树状数组



LOGO

- 形象的表示如图:
  - 每一个结点的权值都是其所有儿子节点权值的和



- 抽象的:

树状数组

可以用一个数组s[]表示;

但s[i]并不是表示a[1]+...a[i]的和,而是a[i-2^k+1]+...+a[i]的和

其中 k 是 i 的二进制表示中末尾0的个数

比如 6 二进制为110,所以其k值为1，s[6]表示a[5]+a[6];

8 二进制为1000; 所以其k值为3，s[8]表示a[1]+...+a[8];

LOGO

- 要想知道s[i]表示的是哪个区域的和,只要求出2^r(即lowbit);
- 树状数组之所以高效简洁的原因就是能够利用**位运算**直接求出i对应的lowbit

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

2^r(lowbit)求法

10100...10111000

&

01011...01001000

= 00000...00001000

LOGO

Why lowbit(x)=x&-x？

- 假设x对应的二进制表达式为a1b，a表示最后的1前面的二进制数码，b全为0或不存在
- -x：x按位取反，末尾加1，

于是

$-x=\sim(a1b)+1=(\sim a)0(\sim b)+1=(\sim a)1b$

- $x\&-x=(a1b)\ \&\ ((\sim a)1b)=(0...0)1(0...0)$

5 的二进制表示：0101；
-5的二进制表示: 1011；

二者相加：

```
  0 1 0 1
+ 1 0 1 1
---------------
1 0 0 0 0
```

LOGO

求和S[k]

```
int sum(int k)
{
	int ret = 0;
	while(k)
	{
		ret += c[k];      //c[k]记录了最后lowbit(k)个数之和
		k -= lowbit(k);  //将k往前移lowbit(k)位，进入前一个子树
	}
	return ret;
}
```

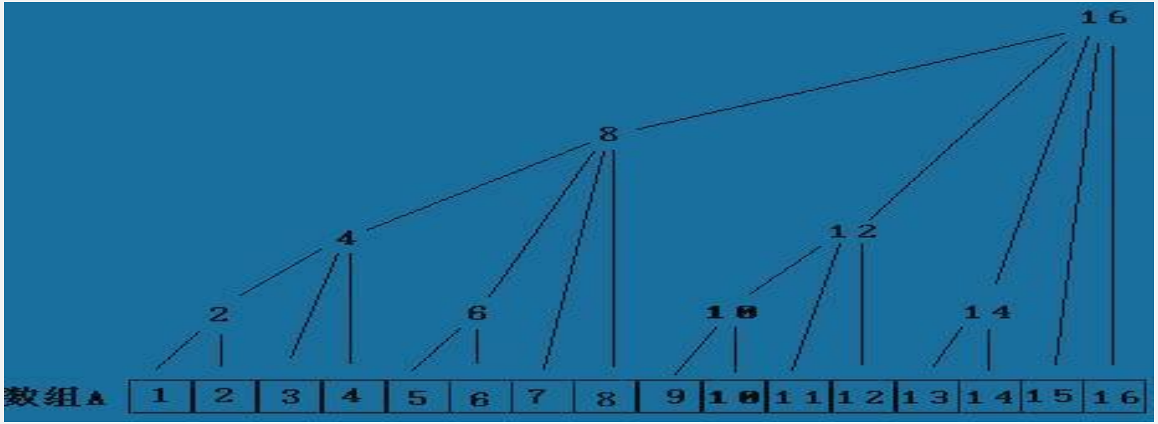
LOGO

求和S[k]

- 每做一次循环，k在二进制表示中会少一个1
- k<=n，k二进制中有O(lgn)位
- 所以一次sum(k)的时间复杂度为O(lgn)

LOGO

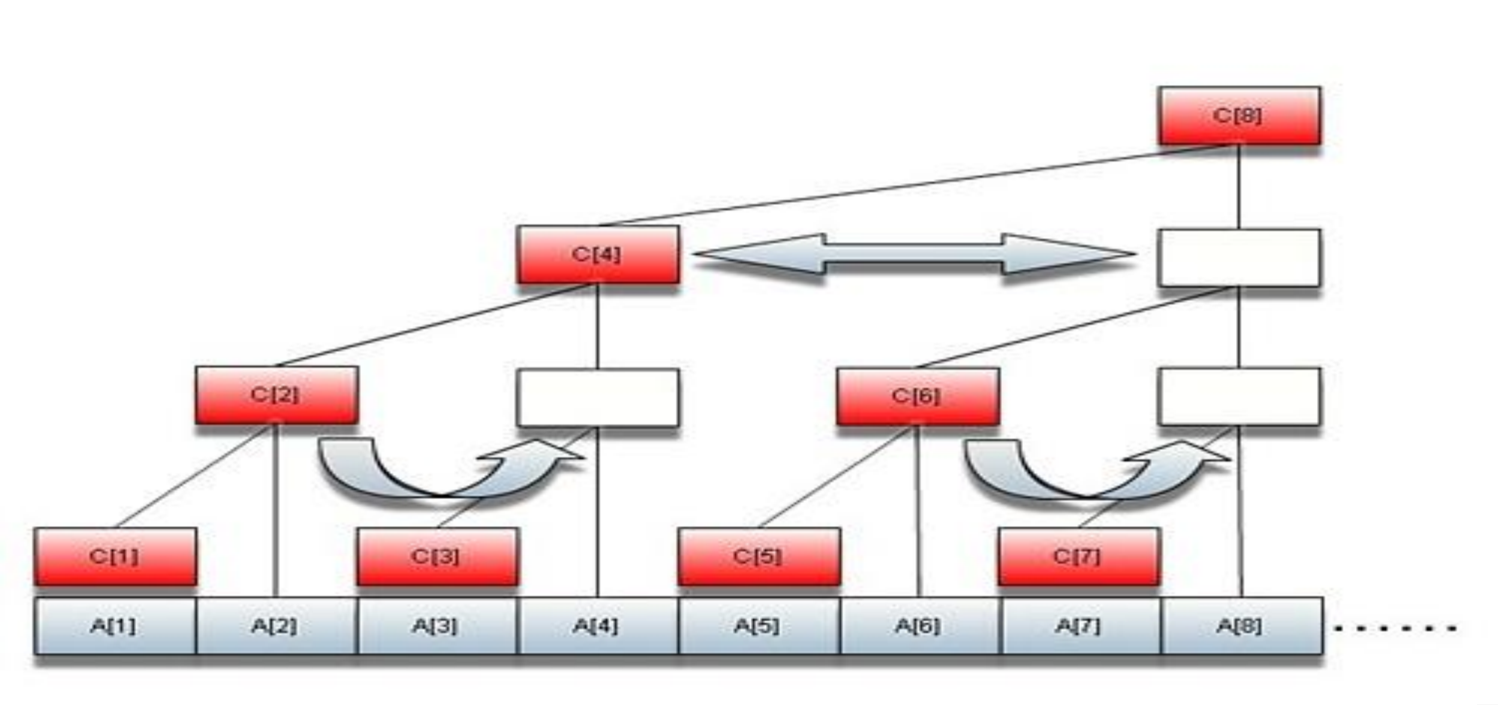
修改modify



- 修改了某个a[i],就需改动所有包含a[i]的s[j];
- 从上图看就是要更改从改叶子节点到根节点路径上的所有s[j]

- 但是怎么求一个节点的父节点呢? Thinking……

LOGO



- 增加虚构点,变成满二叉树!!!
- 每个节点的父亲就跟其右兄弟一样了;
- 而左右兄弟的管辖区域是一样的;
- 所以: i 的父节点就是i+lowb(i);

找父节点

LOGO

修改a[k]

```
int change(int k,int delta)
{
    while(k <= n)
    {
        c[k] += delta;   //修改当前子树的和
        k += lowbit(k);  //当前子树包含数的个数为lowbit(k),
                         // 以它的兄弟节点为根的子树也包含
                         // lowbit(k)个数，k+=lowbit(k)相当于
                         // 把k移到它的父亲节点
    }
    return ret;
}
```

LOGO

修改a[k]

- 每做一次循环，k在二进制表示中最右边的1至少向左移一位
- 最多只能往左移O(lgn)位，所以O(lgn)次循环
- 所以一次change(k,delta)的时间复杂度为O(lgn)

LOGO

回到例题

- 先初始化s[];
- 对于每个C a b执行:modify(a,b);
- 对于每个Q a b输出:sum(b)-sum(a-1);

- 总的时间复杂度为O((c+q)*logn);

LOGO

树状数组

- 求数组前缀和，支持动态的修改操作
- 时空复杂度低，常数小
- 编程复杂度低（优化dp等）
- 可扩展性不好

LOGO

所以，树状数组适合单个元素经常修改而且还反复要求部分的区间的和的情况。

这样的问题虽然也可以用线段树解决，但是用树状数组来做，编程效率和程序运行效率都更高

线段树可以做到的，树状数组不一定能，树状数组可以做到的，线段树一定能。

LOGO

例题

- 乒乓比赛（Ping pong, Beijing 2008, LA4329)
  - 一条大街住着n个乒乓球爱好者，经常组织比赛切磋技术，每个人都有一个不同的技能值a_i。每场比赛需要3个人：两名选手、一名裁判。他们有一个奇怪的规定：裁判必须住在两名选手之间，且技能值也在两名选手之间。问：一共能组织多少种比赛。
  - 输入格式：输入第一行是数据组数T(1≤T ≤20)。每组数据占一行，首先是整数n(3≤n≤20000)，然后是n个不同的整数，即$a_1,a_2,\ldots,a_n(1\le a_i\le 100000)$，按照住所从左到右的顺序给出每个乒乓球爱好者的技能值。
  - 输出格式：对于每组数据，输出比赛总数的值。

例：乒乓比赛的分析

- 考虑第i个人当裁判的情况：假设a_1到a_{i-1}中有c_i个比a_i小，那么就有$(i-1)-c_i$个比a_i大；同理，假设a_{i+1}到a_n中有d_i个比a_i小，那么就有$(n-i)-d_i$个比a_i大。根据乘法原理和加法原理，i当裁判有$c_i(n-i-d_i)+(i-c_i-1)d_i$种比赛。

如何求 c_i 和 d_i？

- 从左到右扫描所有a_i，对于每个i
  - 第i个人当裁判，其等级为a_i，那么比其小的等级有$a_i-1,a_i-2,\ldots,1$
  - 定义数组x[]（x[j]表示目前为止已考虑的所有a_i中是否存在一个等级为j的。x[j]=1表示等级为j的选手已存在）
  - $c_i=x[1]+\ldots+x[a_i-1]$表示a_1到a_{i-1}中比a_i小的个数
- 类似的方法求d_i

乒乓比赛的分析

- 考
  c_i
  大

如何

- 从左到右扫描所有a_i，
  - 第i个人当裁判，其等级为a_i，那 比其小的等级有a_i-1,a_i-2,...,1
  - 定义数组x[]（x[j]表示目前为止已考虑的所有a_i中是否存在一个等级为j的。x[j]=1表示等级为j的选手已存在）
  - c_i=x[1]+...+x[a_i-1]表示a_1到a_{i-1}中比a_i小的个数
- 类似的方法求d_i

使用树形数组：

- 初始x[i]=0，计算c[i]时先使x[ai]=1。
- 树形数组BIT动态修改每个元素的值（即频率），然后求其前缀和（即累计频率）。

——BIT的标准用法

LOGO

二叉索引树

- 二叉索引树（Binary Index Tree， BIT）
- 树形数组
- Peter M. Fenwick于1994年提出， 旨在解决数据压缩里的累计频率(cumulative frequency)计算问题，现多用于高效计算数列的前缀和

$$\sum_{i=1}^{N} a[i]$$

- 特点：O(logn)时间得到上述前缀和；O(logn)时间对某项a[i]添加一个常数

LOGO

二叉索引树受启发于…

- 受启发于：

任何正整数都可以表示成一些2的幂次方，如：

13（二进制表示1101）$=2^0+2^2+2^3$

- 类似地，累计频率可以表示成其子频率集合的和，其中每个集合由不重复的连续频率构成

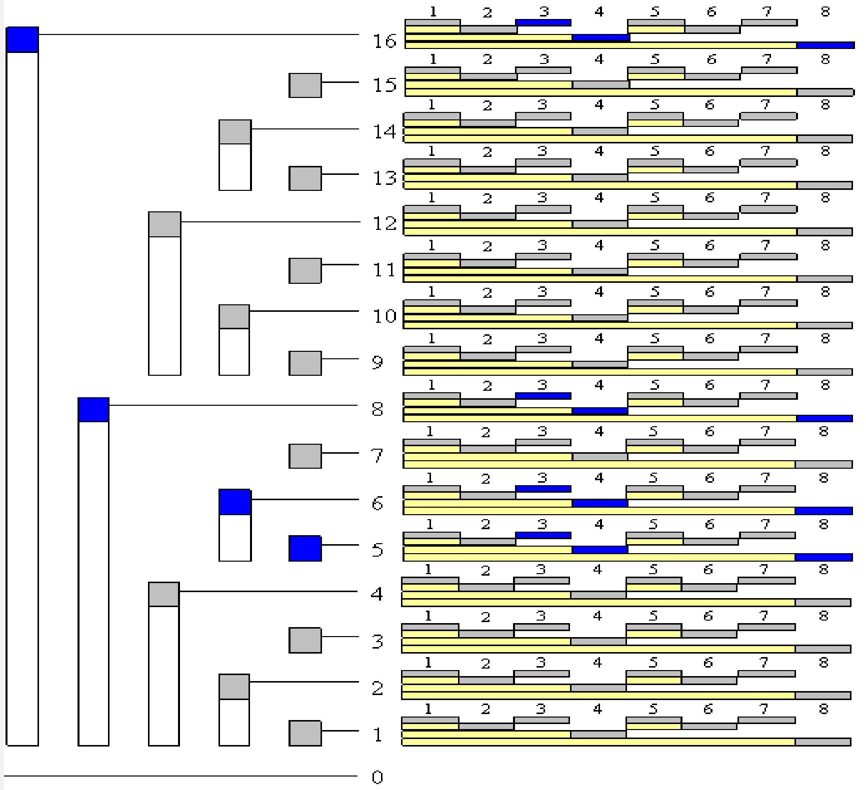
如：

累计频率c[13]=f[1]+f[2]+…+f[13]，可以表示成c[13]=s1+s2+s3+s4,

s1=f[1]+f[2]+…+f[6], s2=f[7]+f[8],

s3=f[9]+f[10] +f[11],s4=f[12]+f[13]

2D二叉检索树更新函数的示意图



其他的函数呢？

n维的情况？

LOGO

Thank you