L32：树状数组

吉林大学计算机学院
谷方明
fmgu2002@sina.com

问题

- 对一个数列，进行两种操作：

1. 将一个数增加 d
2. 求出指定区间的所有数的和

样例输入

5 2
1 2 3 4 5
1 3 2
2 2 4

样例输出

11

- 数据范围：**1≤n,m≤500000**，区间和属于**[−2^{31},2^{31})**;
- 时限: 1s, 空限: 512M

分析

- 单点修改 + 区间求和
- 朴素方法?
  - ✓ 超时
- 前缀和方法？
  - ✓ 超时

区间拆分

- 任何一个正整数x都可拆分成二进制求和的形式

  x = 13 = 0b1101

  = 0b1000 + 0b100 + 0b1

  = 2^3 + 2^2 + 2^0

  = 8 + 4 + 1

- 因此，区间[1,13]可以拆分成长度分别为8、4、1的3个小区间

  [1,13] = [1,8] + [9,12] + [13,13]

  // 0b1000 + 0b100 + 0b1

lowbit函数

- 拆成多少个小区间，取决于 x 的二进制表示中有多少个 1；每个小区间的长度，就是相应位上的 1 代表的数。
- lowbit(x): x的二进制表示中最低位的 1 代表的数

  lowbit(13) = lowbit(0b1101) = 1

  lowbit(12) = lowbit(0b1100) = 4

  lowbit(8)   = lowbit(0b1000) = 8

- 所有的小区间的长度，可以不断执行 lowbit 函数来得到

区间拆分实现

- 每个小区间的右端点是 x 的话，长度就是 **lowbit(x)**

```
while (x) {
    // 当前小区间的右端点是 x
    // 当前小区间的长度是 lowbit(x)
    // 然后减去它，继续
    x -= lowbit(x);
}
```

- 原区间 [1,x] 可以拆成 log(x) 个小区间；

lowbit函数的实现

- 设lowbit(x) = k,
  - ✓ x： x的第k位为1，后续都为0
  - ✓ ~x (各位取反)： ~x的第k位为0，后续都为1
  - ✓ ~x + 1：~x第k位后的1因进位都变成0，而第k位因进位变成1
  - ✓ x & (~x+1): 第k位是1，其它位均为0；
  - ✓ 计算机编码： -x = (~x+1)   //补码
- **lowbit(x) = x & (~x + 1) = x & -x**

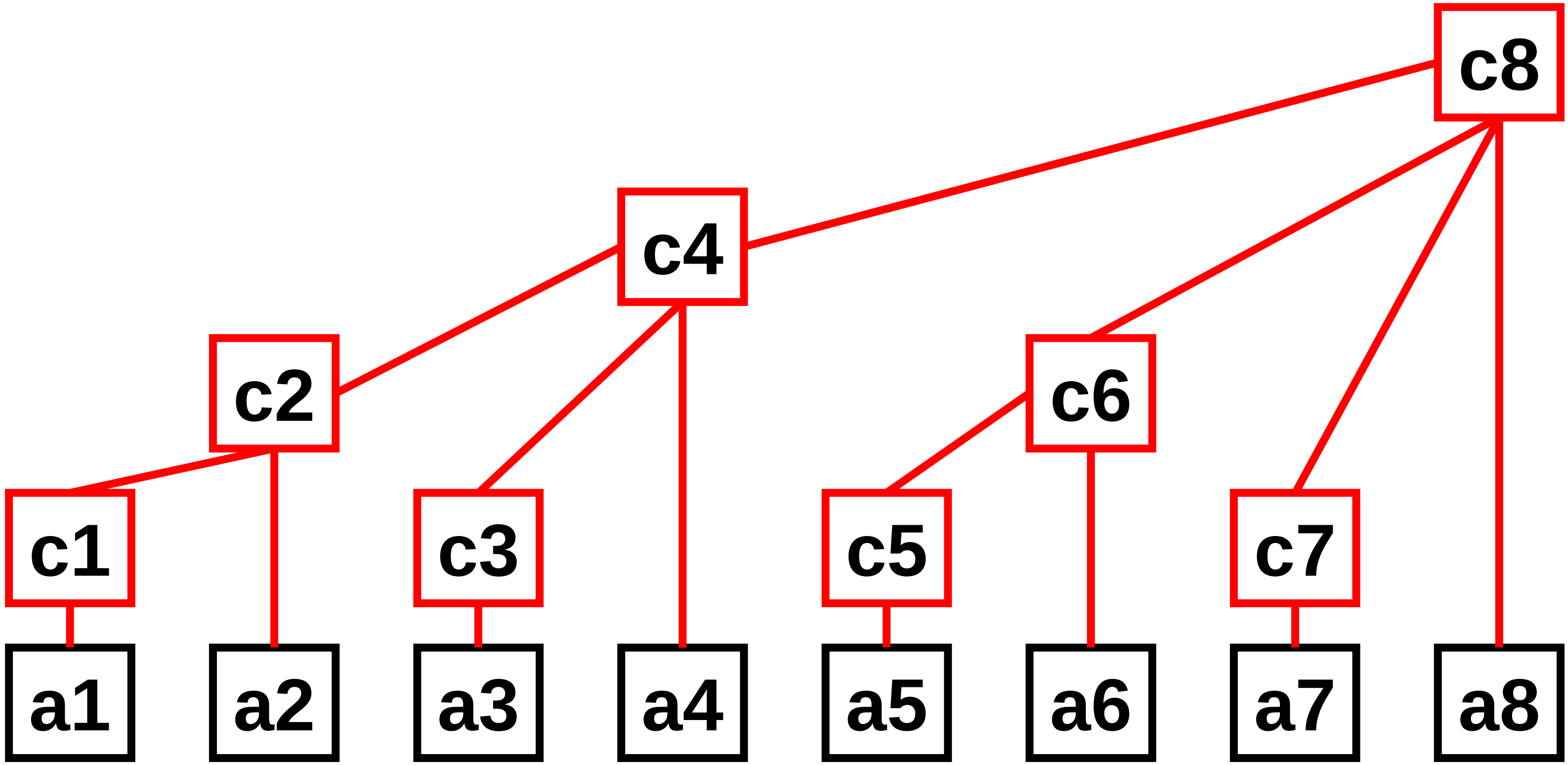
树状结构

- 定义数组c
  - ✓ c[x] 表示拆分成的以 x 为右端点的小区间的和
  - ✓ 即c[x] 维护原数组 a[x] 在区间 [x-lowbit(x)+1, x] 上的和

- 如果一个小区间完全覆盖另一个小区间，那么前者可以称为后者的祖先。最近的祖先就是父结点。

树状数组

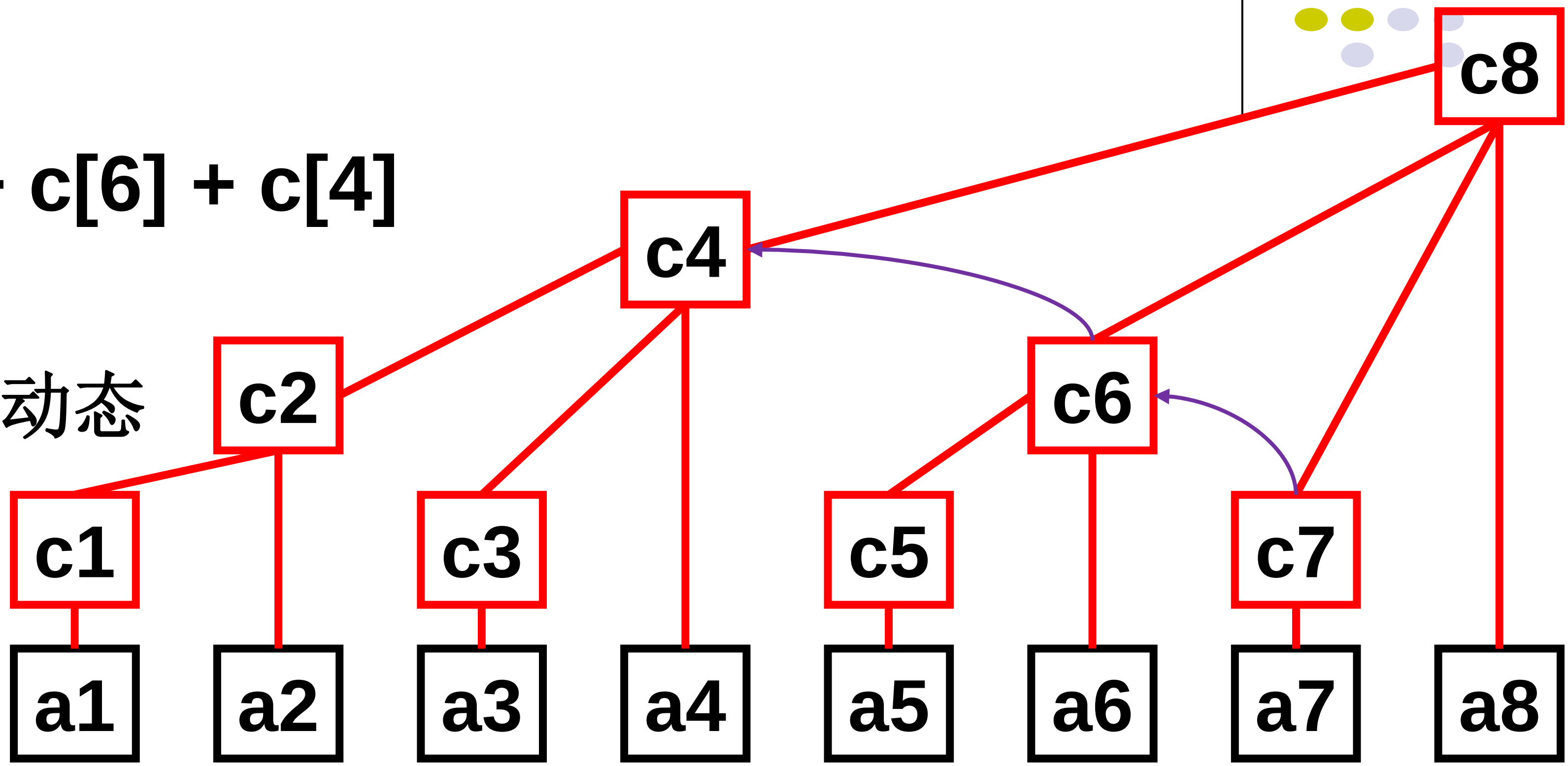
- 性质
  - ✓ 父结点 c[x] 保存其所有儿子结点的总和
  - ✓ 结点 c[x] 的父结点是 c[x+lowbit(x)]
  - ✓ x 为 2 的幂时，c[x] 会管辖整个前缀区间 (lowbit 是自身）



前缀区间求和

- 例 sum[1,7] =c[7] + c[6] + c[4]
- 参考代码

```
int c[MAXN+1]; //全局或动态
// 查询区间 [1, x] 上的和
int sum(int x) {
    int ans = 0;
    for (; x; x -= lowbit(x)) ans += c[x];
    return ans;
}
```



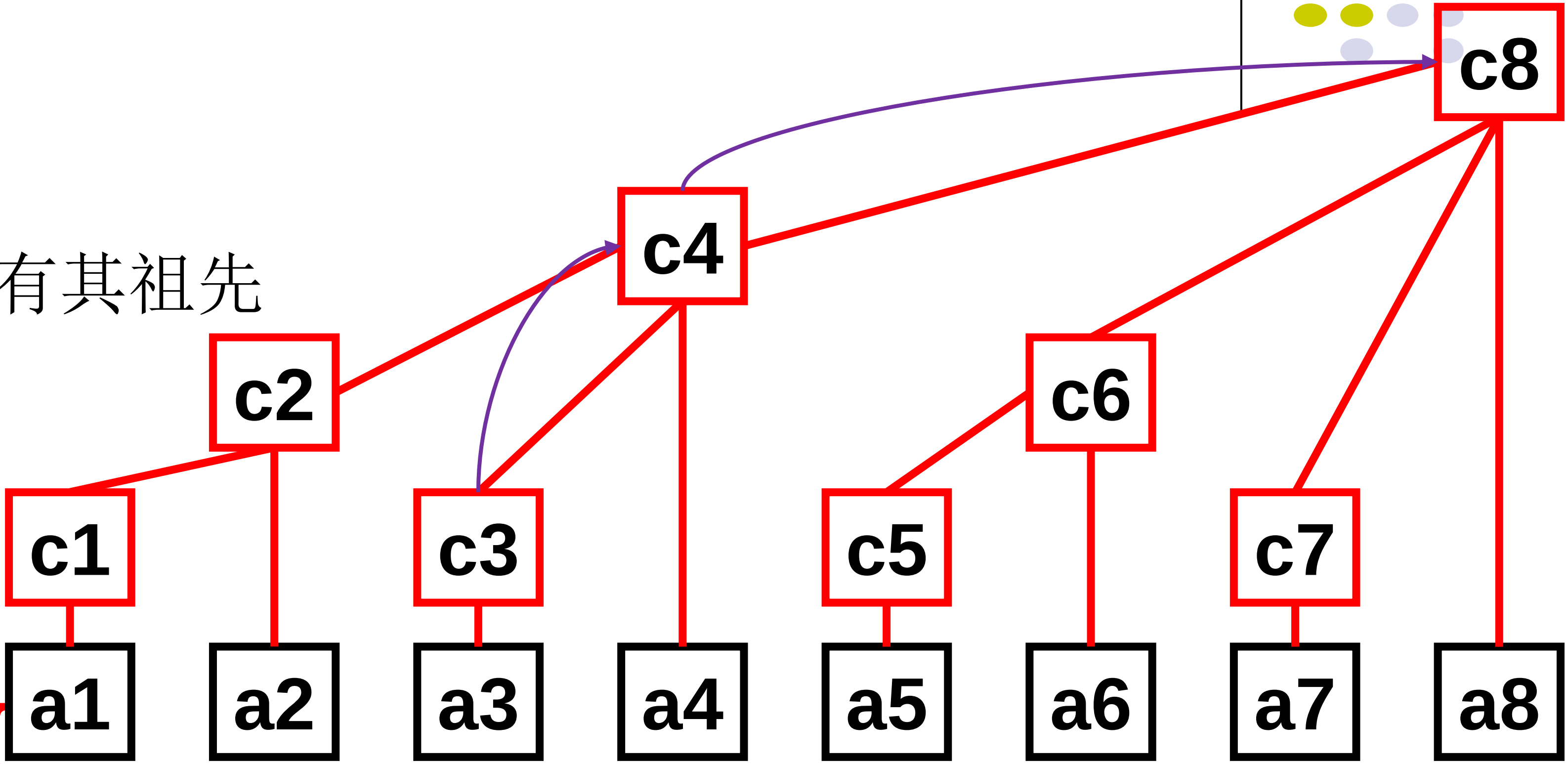
区间求和

- 树状数组下标从1开始; c[0] = 0;

- query[a,b] = sum[b] – sum[a-1];

单点修改

- 例：**add(3, d)**
  - ✓ 包含a[3]的小区间，只有其祖先



- 参考代码

```
void add(int x, int d) {
    for (; x <= n; x += lowbit(x)) c[x] += d;
}
```

整体参考代码

```
#define lowbit(x) (x & -x)
inline void add(int x,int d) {
    for(; x<=n; x += lowbit(x)) c[x]+=d;
}
inline int sum(int x) {
    int ans=0;
    for(; x; x -= lowbit(x)) ans+=c[x];
    return ans;
}
```

```
int main() {
    int i,op,x,k;
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++) {
        scanf("%d",a+i);
        add(i,a[i]); //建立c
    }
    while(m--) {
        scanf("%d %d %d",&op,&x,&k);
        if(op==1) add(x,k);
        else printf("%d\n",sum(k)-sum(x-1));
    }
   return 0;
}
```

小结

- **时间效率**
  - ✓ 单点修改：最多加logn个二进制位，log(n)
  - ✓ 区间查询：最多减logx个二进制位，log(n)

- **树状数组又称二叉索引树（Binary Indexed Tree，BIT）或Fenwick树。**
  - ✓ Peter M. Fenwick1994年发表文章：A New Data Structure for Cumulative Frequency Tables。
  - ✓ 初衷是解决数据压缩里的累积频率的计算问题，现多用于高效计算数列的前缀和、区间和等。

拓展

- **支持区间修改、单点查询的树状数组**
  - ✓ 利用差分数组，a[0] = 0;
  - ✓ 区间修改[i,j] : c[i] +=d，c[j+1] -=d,
  - ✓ 单点查询x： sum(x)

- **特点**
  - ✓ 实现简单；
  - ✓ 本质是维护前缀和（加/乘），区间查询通过减（减/除）运算完成；
  - ✓ 更一般的区间和（最值、gcd等），不易支持，要求助线段树等工具