线段树

吉林大学计算机学院
谷方明
fmgu2002@sina.com

例1 区间和查询

- 输入N个数和M次操作，操作有两种：

1. 格式为1 x y,表示把第x个数加y
2. 格式为2 L R,每次询问L到R之间所有数的和。

- N,M<=100000

分析

- 暴力
  - ✓ 每次修改O(1)
  - ✓ 每次查询O(N)
  - ✓ M次操作最坏O(MN)

- 更好的方法?

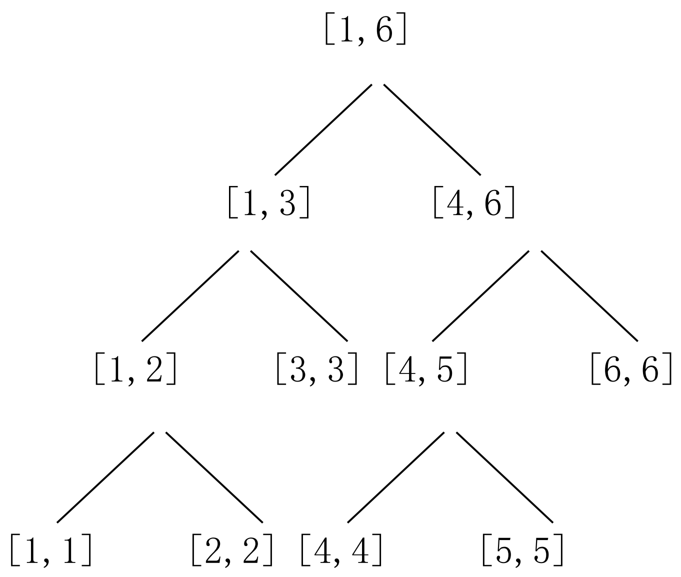
线段树的定义

- 线段树是一棵二叉树，结点表示一个区间。

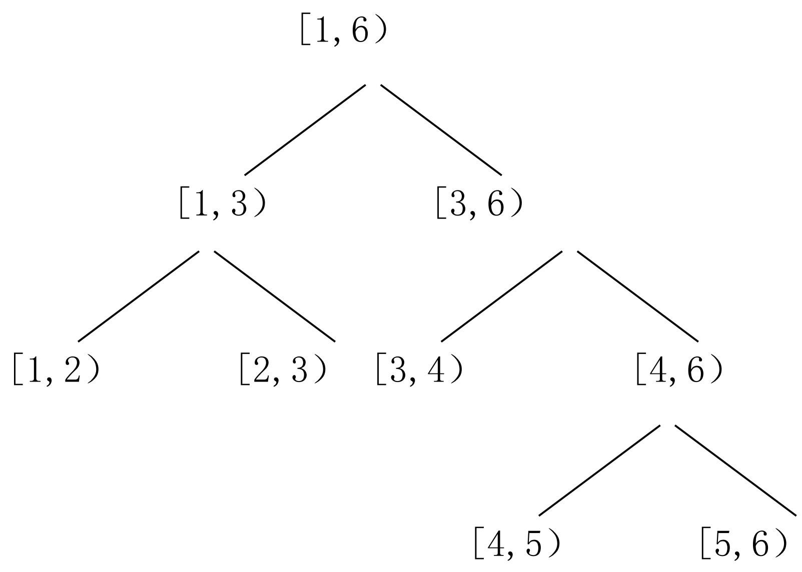
1. 根结点表示整个区间[1,n]
2. 如果某个结点区间[L,R]不能再分，则该结点为叶节点
3. 否则以[L,(L+R)/2]构造出左子树,[(L+R)/2+1,R]构造出右子树

- 说明
  - ✓ 线段树记为T(a, b)，表示区间[a,b]；b-a称为区间的长度，记为L。
  - ✓ 用户可根据需要，对区间表示进行修改，如[a,b)。
  - ✓ 定义是递归的，操作可用递归实现。

线段树两种表示方法



多用于数列，也称点树



多用于线段，也称线树

线段树的性质

1. **线段树的结点数为2n-1**

- **证明：**
  - ✓ 线段树不存在度为1的结点
  - ✓ 叶结点数为n
  - ✓ 内结点数为n-1

线段树的性质

2. **线段树的高度为⌊log(n-1)⌋+ 1. (或⌊ log2L ⌋;叶结点高度为0)**

- 证明：
  - ✓ $2^{h}-1+2 \leqslant 2n-1 < 2^{h+1} -1 + 2$
  - ✓ $\log(n-1) < h \leqslant \log(n-1) + 1$

线段树的性质

3. 线段树T(a,b)把区间上的任意一条线段都分成不超过**2log*L***条线段

- 证明参考附录

- 这些性质为线段树能在$O(\log n)$的时间内完成一个区间的插入、删除、查找等工作，提供了理论依据

线段树的存储

- 链式存储
  - 直观自然
  - 缺点：动态申请等导致效率略低
  - 一般用静态链表实现
- 顺序存储
  - 线段树除了最底层是一棵满二叉树。
  - 完全二叉树编号最大多少？不超过4*n
- 压缩存储
  - 改变编号方案

线段树存储——静态链表版

```
struct Node{
        int l,r,sum;
        int lson,rson;
};
Node seg[2*MAXN];
int root, sp;

int n,a[MAXN];
```

线段树操作——建树

```
void build(int cur, int l, int r) {
    int mid = (l+r)/2;
    seg[cur].l = l;  seg[cur].r = r;
    if(l==r){
        seg[cur].sum = a[l];
    }else{
        seg[cur].lson = ++sp; seg[cur].rson = ++sp;
        build(seg[cur].lson,l,mid);  build(seg[cur].rson,mid+1,r);
        seg[cur].sum = seg[seg[cur].lson].sum + seg[seg[cur].rson].sum;
    }
}   //调用make(root,1,n),时间复杂度为O(N)
```

线段树操作——区间查询

```
int query(int cur, int l,int r) {
        int ans = 0,mid=(seg[cur].l + seg[cur].r)/2;
        if(l<=seg[cur].l && r>=seg[cur].r)
            return seg[cur].sum;
        if(l <= mid)
            ans += query(seg[cur].lson,l,r);
        if(r > mid)
           ans += query(seg[cur].rson,l,r);
        return ans;
}  //时间复杂度O(logn)
```

线段树操作——单点修改

```
void change(int cur, int x, int delta){
    int mid = (seg[cur].l+seg[cur].r)/2;
    if(seg[cur].l == seg[cur].r)
        seg[cur].sum+=delta;
    else{
        if(x <= mid) change(seg[cur].lson,x,delta);
        if(x > mid) change(seg[cur].rson,x,delta);
        seg[cur].sum=seg[seg[cur].lson].sum+seg[seg[cur].rson].sum;
    }
} //时间复杂度O(logn)
```

例2:

- 输入N个数和M次操作，操作有两种：

1. 格式为1 L R y,表示把L到R之间的所有数都加y
2. 格式为2 L R,每次询问L到R之间所有数的和。

- N,M<=100000

分析

- **如果用单点修改完成区间修改，时间复杂度为O(nlogn)。**

- **lazy-tag技术（延迟修改）**
  - ✓ 将修改标记标注在区间上；而不是直接修改所有叶结点。
  - ✓ 如果后续的维护或查询过程中，需要对这个结点进行递归处理，就当场把这个标记分解，传递给它的两个孩子结点；
  - ✓ 这种在需要时才进行分解传递的技术，使整体处理的时间复杂度仍为O(logn)

线段树操作——区间修改

- //Node增加一个delta域，代表该区间的修改量

```
void change(int cur, int l,int r, int delta){
    int mid = (seg[cur].l+seg[cur].r)/2;
    if( l<=seg[cur].l && r>=seg[cur].r ){
        seg[cur].sum += delta * (seg[cur].r-seg[cur].l+1);
        seg[cur].delta += delta;
    } else{
        if(seg[cur].delta != 0) update(cur);
        if(l <= mid) change(seg[cur].lson,l,r,delta);
        if(r > mid) change(seg[cur].rson,l,r,delta);
        seg[cur].sum = seg[seg[cur].lson].sum + seg[seg[cur].rson].sum;
    }
```

线段树操作——延迟信息下传

```
void update(int cur){
        seg[seg[cur].lson].sum += seg[cur].delta*(seg[seg[cur].lson].r-seg[seg[cur].lson].l+1);
        seg[seg[cur].rson].sum += seg[cur].delta*(seg[seg[cur].rson].r-seg[seg[cur].rson].l+1);

        seg[seg[cur].lson].delta +=  seg[cur].delta;
        seg[seg[cur].rson].delta +=  seg[cur].delta;
        seg[cur].delta = 0;
}
```

线段树操作——区间查询

```
int query(int cur, int l,int r){
        int ans = 0,mid=(seg[cur].l + seg[cur].r)/2;
        if(l<=seg[cur].l && r>=seg[cur].r) return seg[cur].sum;
        if(seg[cur].delta != 0) update(cur);
        if(l <= mid)
            ans += query(seg[cur].lson,l,r);
        if(r > mid)
            ans += query(seg[cur].rson,l,r);
        return ans;
}
```

线段树的适用条件

- 建模成数轴上或序列上的区间问题，通常支持如下操作：

1. 统计操作: 要求具有可加性。例如
   - ✓ 数字之和——总数字之和 = 左区间数字之和 + 右区间数字之和
   - ✓ 最大公因数(GCD)——总GCD = gcd(左区间GCD , 右区间GCD);
   - ✓ 最大值——总最大值=max(左区间最大值，右区间最大值)
2. 修改操作: 对数轴上的一个区间或数列中的连续若干个数进行一种相同的处理。

线段树的拓展和应用

- 非递归实现
- 离散化
- 多维
- **扫描线**
- **持久化（主席树）**

附录：定理2证明

- 线段树把区间上的任意一条线段都分成不超过2log*L*条线段
- 证明(1)：在区间(*a*，*b*)中，对于线段(*c*，*d*)，如果(*c*<=*a*) 或 (*d*>=*b*)，那么线段在 (*a*，*b*)中被分为不超过log(*b*-*a*)。
- 用归纳法证明。
- 如果是单位区间，最多被分为一段，成立。
- 如果区间(*a*，*b*)的左儿子与右儿子成立，
- 那么如果当*c*<=*a*时，若*d*<=(*a*+*b*)div2那么相当与其左儿子分该线段，所分该线段数树不超过log((*a*+*b*)div 2-*a*)，即不超过log(*b*-*a*),成立。若*d*>(*a*+*b*) div 2那么相当于该线段被分为它左儿子表示的线段，加上右儿子分该线段，线段数不超过1+log(*b*-(*a*+*b*) div 2),也不超过log(*b*-*a*)，成立。
- 对于*d*>=*b*的情况证明类似。

- 证明(2)在区间(*a*,*b*)中，对于任意线段也成立。
- 用归纳法证明。
- 对于单位区间，最多分为一段，成立。
- 若(*a*，*b*)的左儿子与右儿子均成立，则对于线段(*c*，d)
- 若d<=(*a*+*b*)div 2 则该区间所分该线段等于其左儿子区间所分该线段，线段数小于log((*a*+*b*) div 2-*a*)<2log(*b*-a)，成立。
- 若*c*>(*a*+b) div 2 则该区间所分该线段等于其右儿子区间所分该线段，线段数小于log(*b*-(*a*+*b*) div 2)<2log(*b*-a)，成立。
- 若1、2均不成立，则此线段在左儿子区间分该线段满足*d*>V.Lson.*b*,分该线段数不超过log(*b*-(*a*+*b*) div 2)，而在右儿子区间分该线段满足 *c*<=V.Rson.*a*,分该线段不超过log((*a*+*b*) div 2-1)，所以在该区间分该线段不超过2log(*b*-*a*)，成立。