

# C++算法

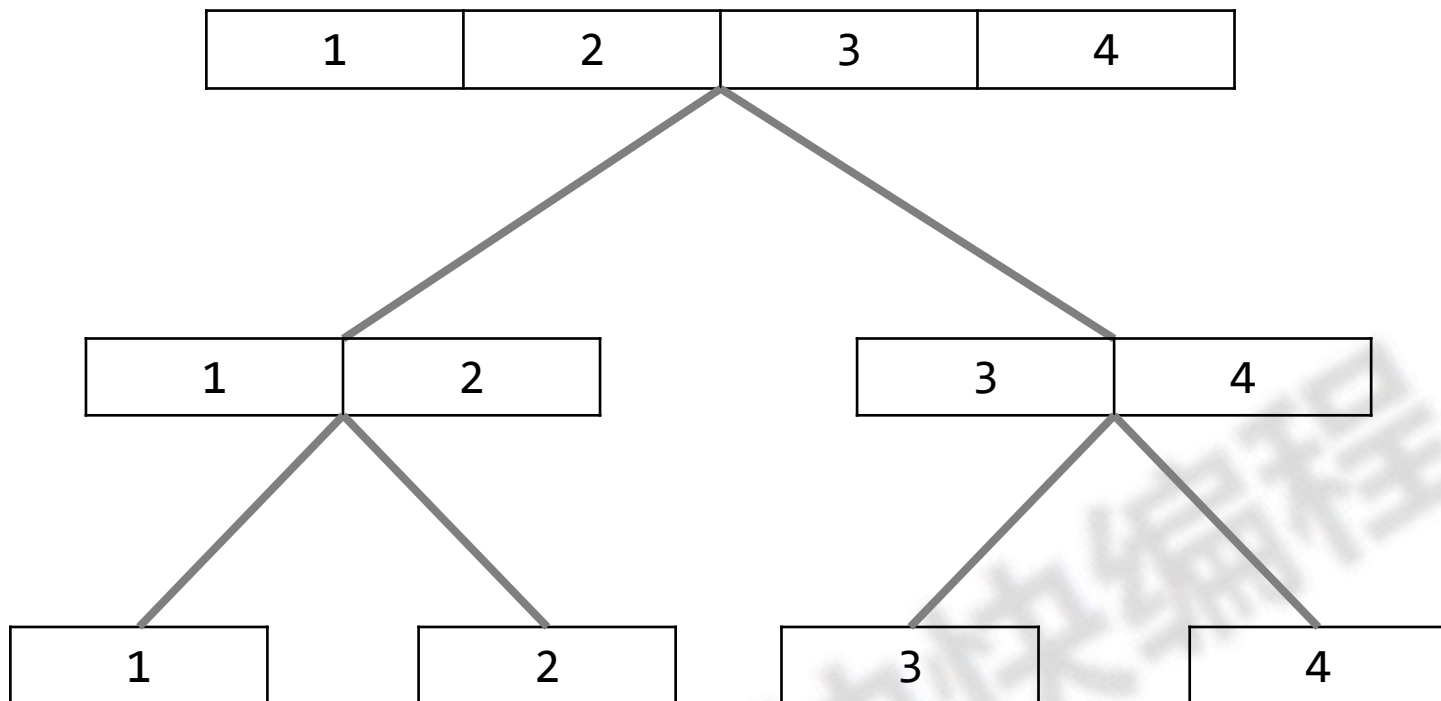
# 线段树

Segment Tree

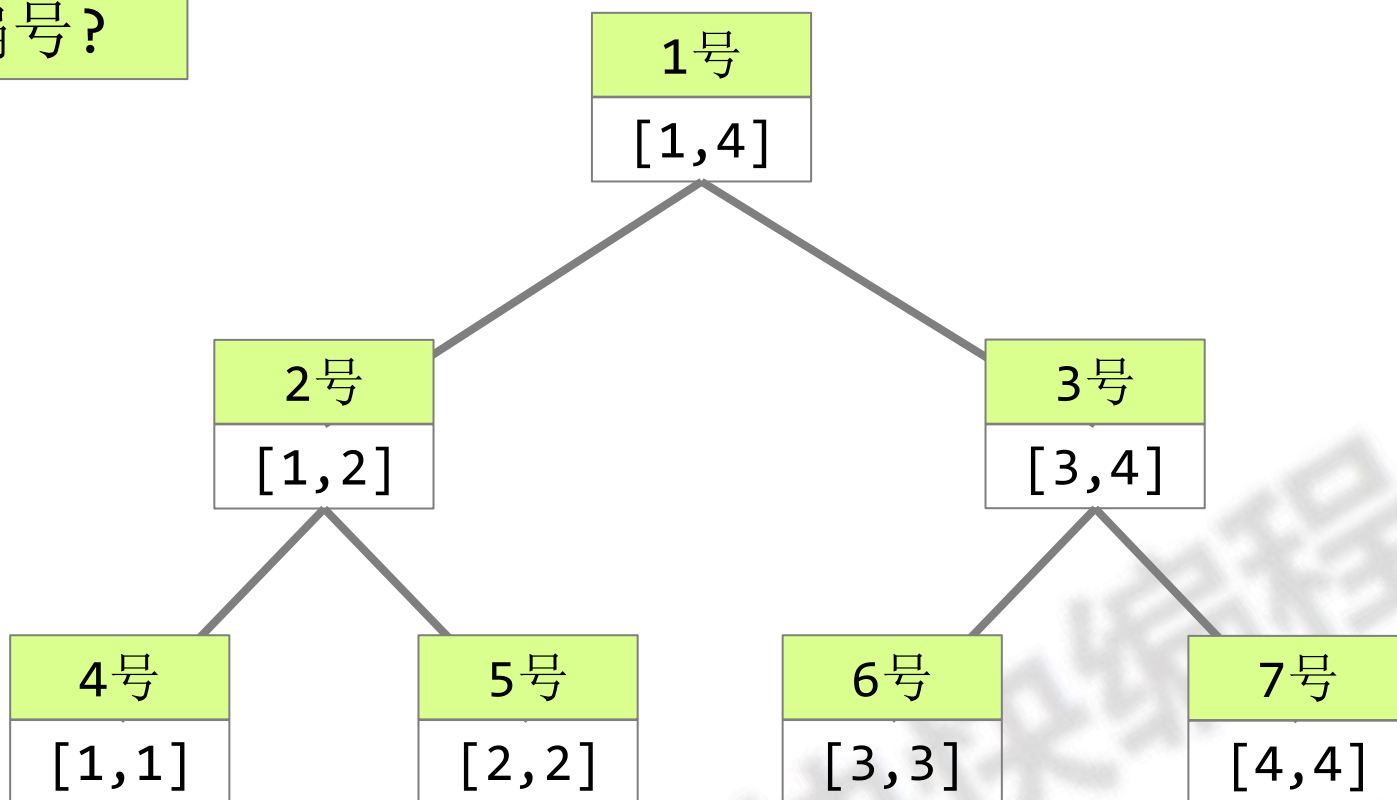
基础应用：RSQ, RMQ

二叉树每个节点对应一个区间

序列连续编号

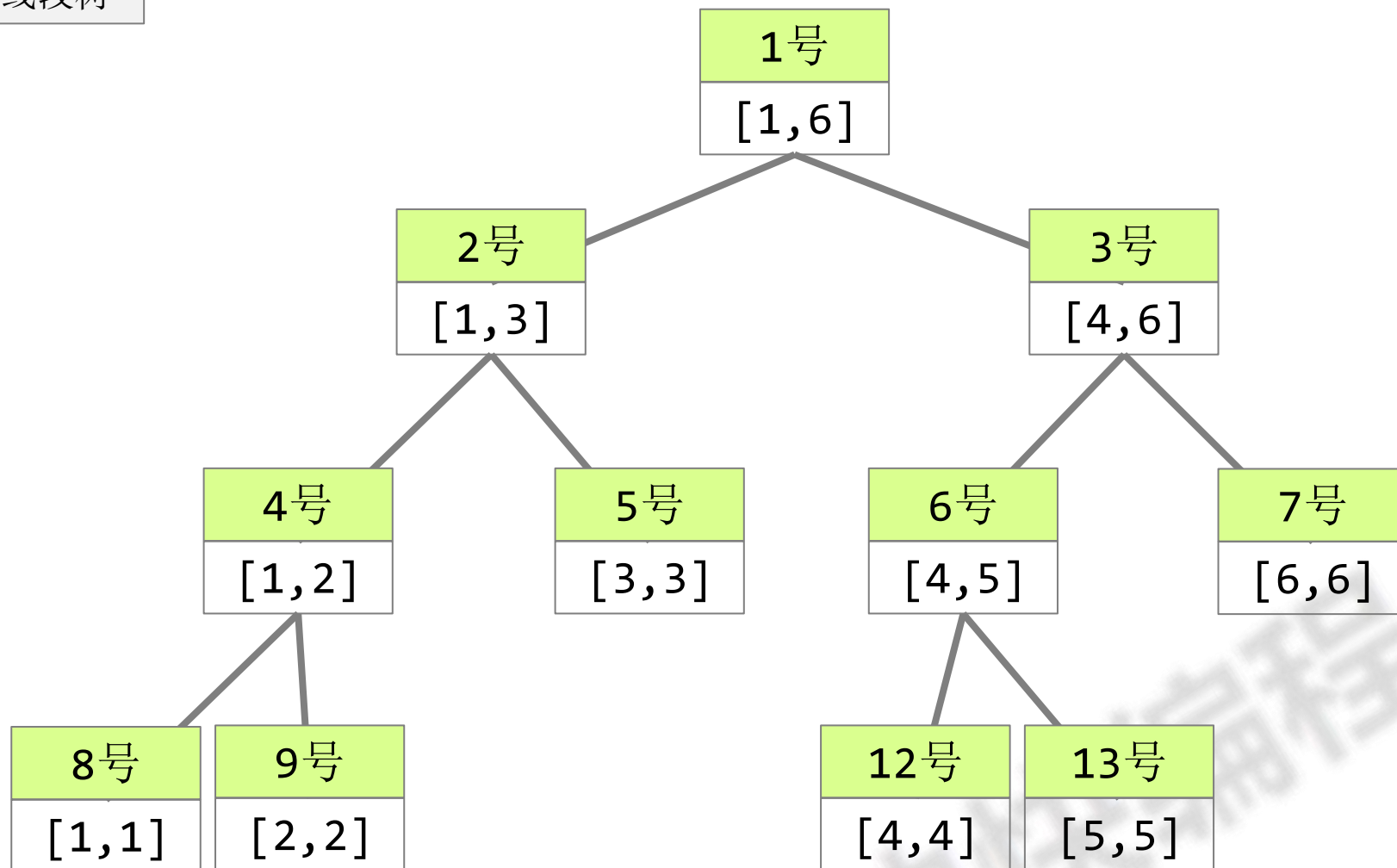


如何为二叉树  
节点编号?



序列长度为4  
线段树节点共7个  
最大节点编号7

## 线段树



序列长度为6  
线段树节点共11个  
最大节点编号13

线段树并不是完全二叉树

线段树对应序列长度为 $n$

线段树深度多少？

$O(\log n)$

线段树对应序列长度为 $n$

提出猜想

线段树共几个节点？

$$2n-1$$

如何证明

每个叶节点对应原序列元素

共 $n$ 个

每个非叶节点对应一次区间切割

共 $n-1$ 个

## 线段树对应序列长度为n

命题

线段树最大编号 $\leq 4n$

证明

假设n在两个2的整数次方之间  
 $2^{k-1} < n \leq 2^k$

长度为n的线段树最大编号  
 $\leq$  长度为 $2^k$ 的线段树最大编号  
 $\leq 2 * 2^k$   
 $\leq 4 * 2^{k-1}$   
 $\leq 4n$



## 线段树节点类型模板

```
struct Segment{  
    ll l;  
    ll r;  
    ll sum;  
};
```

```
struct Segment{  
    ll l;  
    ll r;  
    ll mx;  
};
```

```
Segment tr[N*4];
```

线段树用数组tr储存4N个节点

隐性父子关系 由数组编号对应  
无需额外储存父子连边

## 快快编程676

静态

RMQ

```
struct Segment{  
    ll l;  
    ll r;  
    ll mx;  
};  
Segment tr[N*4];
```

tr[iSeg].mx 维护  
[ tr[iSeg].l , tr[iSeg].r ]  
区间的RMQ

## 建立线段树里iSeg号节点为根的子树 对应原序列[1,r]区间

```
13 void build(ll iSeg, ll l, ll r){  
14     if(l==r){  
15         tr[iSeg]=(Segment){l,r,x[l]};  
16         return;  
17     }  
18     ll m=(l+r)/2;  
19     build(iSeg*2,l,m);  
20     build(iSeg*2+1,m+1,r);  
21     tr[iSeg]=(Segment){l,r,max(tr[iSeg*2].mx,tr[iSeg*2+1].mx)};  
22 }
```

创建叶节点后返回

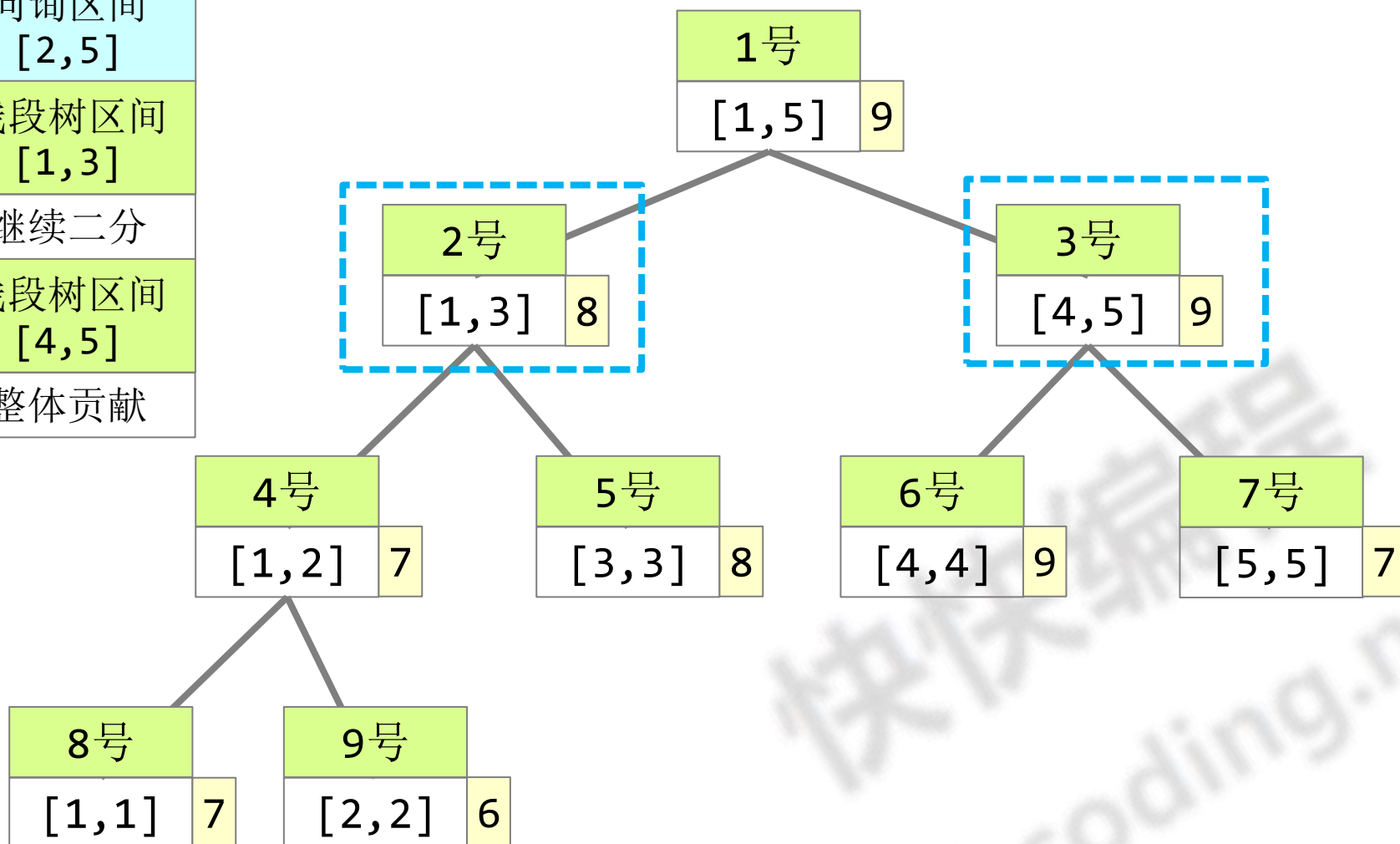
先创建左子树  
再创建右子树

合并RMQ

tr[iSeg].mx 维护  
[ tr[iSeg].l , tr[iSeg].r ]区间的RMQ

i=1	i=2	i=3	i=4	i=5
7	6	8	9	7

询问区间 [2,5]
线段树区间 [1,3]
继续二分
线段树区间 [4,5]
整体贡献



两种区间

线段树区间

操作区间

三种情况

隐含四种情况

线段树区间对操作区间无贡献

线段树不再深入, 返回

线段树区间对操作区间整体贡献

线段树不再深入, 返回

线段树区间对操作区间部分贡献

线段树继续深入, 二分

属于哪一种?

## 计算线段树里iSeg号节点为根的子树 对原序列[1,r]区间rmq的贡献

```
23 11 rmq(l1 iSeg, l1&l, l1&r){  
24     if(r<tr[iSeg].l||tr[iSeg].r<l)return -INF;  
25     if(l<=tr[iSeg].l&&tr[iSeg].r<=r)return tr[iSeg].mx;  
26     return max(  
27 }
```

三种情况  
分类讨论

线段树节点区间对操作区间  
无贡献  
整体贡献  
部分贡献

## 快快编程698

动态

点增加 + RSQ

```
struct Segment{  
    ll l;  
    ll r;  
    ll sum;  
};  
Segment tr[N*4];
```

tr[iSeg].sum 维护  
[ tr[iSeg].l , tr[iSeg].r ]  
区间的RSQ

对线段树里*iSeg*号节点为根的子树更新  
原序列*id*号元素增加*delta*

```
19 void add(ll iSeg, ll&id, ll&delta){  
20     tr[iSeg].sum=(tr[iSeg].sum+delta)%MOD;  
21     if [ ] return;  
22     if(id<=tr[iSeg*2].r)  
23         add(iSeg*2, id, delta);  
24     else  
25         [ ]  
26 }
```

RSQ直接更新  
无需合并区间



## 计算线段树里iSeg号节点为根的子树 对原序列[1,r]区间rsq的贡献

```
27 11 rsq(11 iSeg, 11&l, 11&r){  
28     if(r < tr[iSeg].l || tr[iSeg].r < l) return 0;  
29     if(1 <= tr[iSeg].l && tr[iSeg].r <= r)  
30         return tr[iSeg].sum;  
31     return (rsq(iSeg*2, l, r) + rsq(iSeg*2+1, l, r))%MOD;  
32 }
```

三种情况  
分类讨论

线段树节点区间对操作区间  
无贡献  
整体贡献  
部分贡献




## 快快编程1713

动态

点替换 + RMQ

```
struct Segment{  
    ll l;  
    ll r;  
    ll mx;  
};  
Segment tr[N*4];
```

对线段树里*iSeg*号节点为根的子树更新  
原序列*id*号元素数值替换为*val*

```
23 void update(ll iSeg, ll&id, ll&val){
24     if(tr[iSeg].l==tr[iSeg].r){
25         
26         return;
27     }
28     if(id<=tr[iSeg*2].r)
29         update(iSeg*2, id, val);
30     else
31         
32     
33 }
```

合并RMQ

## 线段树对应序列长度为n

### 提出猜想

单次区间查询涉及到  
最多几个线段树节点

$O(\log n)$

### 如何证明

随着查询深入, 线段树节点区间越分越短

区间查询分两个阶段

1. 线段树节点区间完全包含查询区间
2. 线段树节点区间不完全包含查询区间  
二分切割后至少一个线段树子节点会终止深入访问（无贡献/整体贡献）

# 快快编程

676, 698, 1713

性质证明发课程群

拓展题

126, 243