线段树的合并与分裂

主讲人：邓哲也

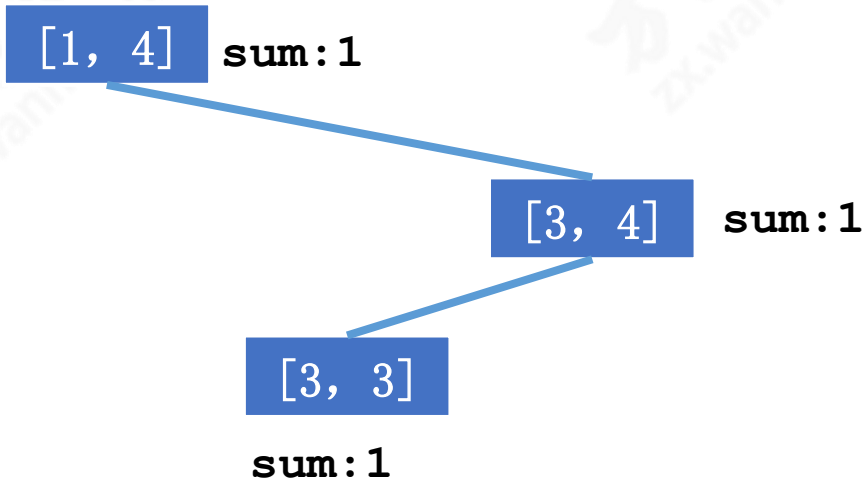
动态开节点的线段树

有时候为了节省内存，我们不需要一开始就建好整颗线段树，而是每次要修改某个位置的值的时候再去新建。

比如我们一开始只有一个［1，4］节点。

动态开节点的线段树

然后把 [3, 3] 加一。



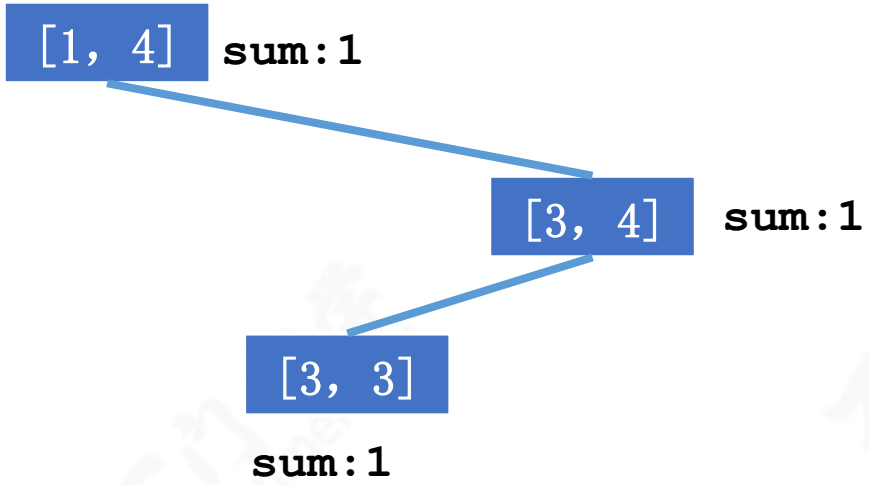
动态开节点的线段树

此时我们一样能执行询问操作。

查询［3，4］会返回 1

查询［4，4］会找不到需要查的节点，返回 0

查询［1，3］，左半部分找不到需要查的节点，返回 0；右半部分会查到 1.



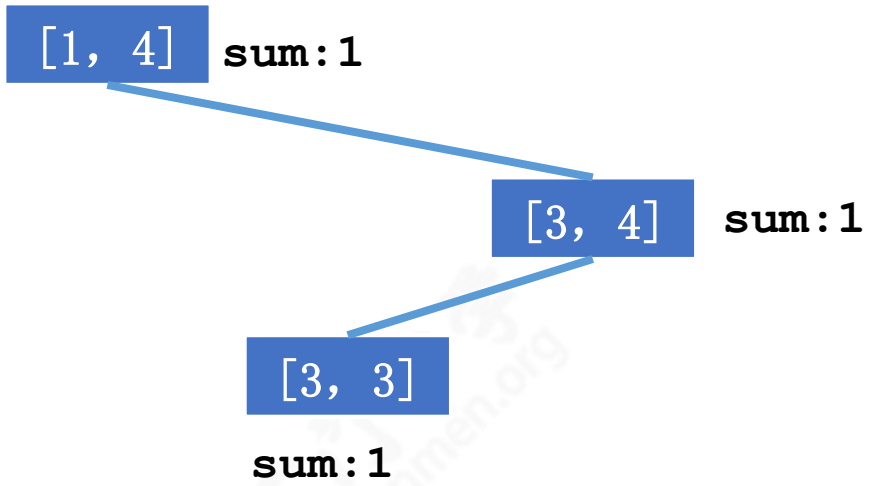
动态开节点的线段树

此时我们一样能执行询问操作。

查询［3，4］会返回 1

查询［4，4］会找不到需要查的节点，返回 0

查询［1，3］，左半部分找不到需要查的节点，返回 0；右半部分会查到 1.



动态开节点的线段树

我们用 lc[x]，rc[x] 来表示 x 的左右子节点。

```
void update(int x){
    sum[x] = sum[lc[x]] + sum[rc[x]];
}
```

动态开节点的线段树

我们用 lc[x]，rc[x] 来表示 x 的左右子节点。

```
void modify(int p, int v, int l, int r, int &x){
    if (!x) x = ++ tot;
    int mid = (l + r) >> 1;
    if (p <= mid) modify(p, v, l, mid, lc[x]);
    else modify(p, v, mid + 1, r, rc[x]);
    update(x);
}
```

动态开节点的线段树

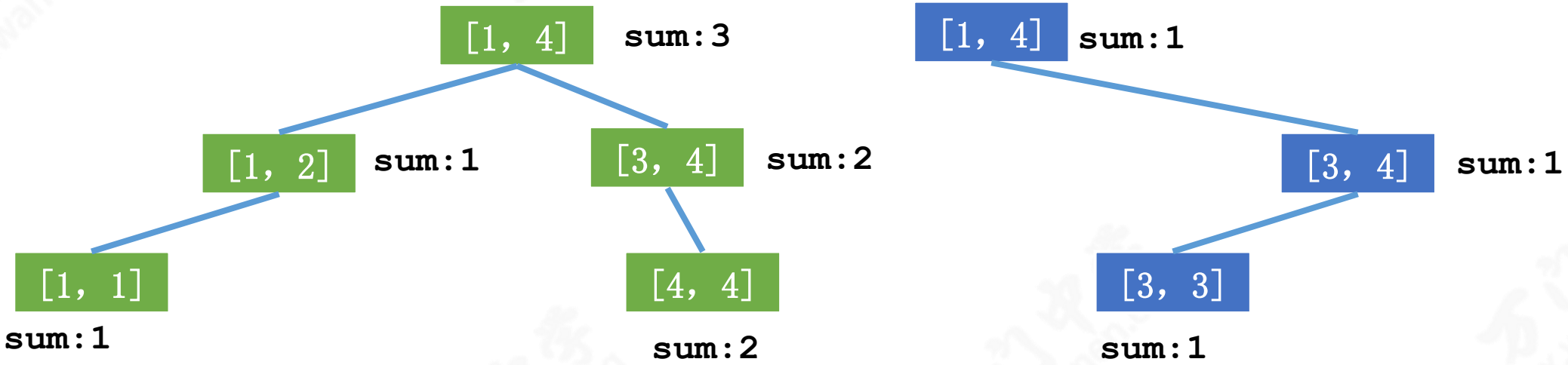
我们用 lc[x]，rc[x] 来表示 x 的左右子节点。

```
int query(int A, int B, int l, int r, int x){
    if(!x) return 0;
    if (A <= l && r <= B) return sum[x];
    int mid = (l + r) >> 1, ret = 0;
    if (A <= mid) ret += query(A, B, l, mid, lc[x]);
    if (mid < B)  ret += query(A, B, mid + 1, r, rc[x]);
    return ret;
}
```

线段树的合并

当线段树维护的权值范围［1，n］是一样的时候，线段树的形态是唯一的。

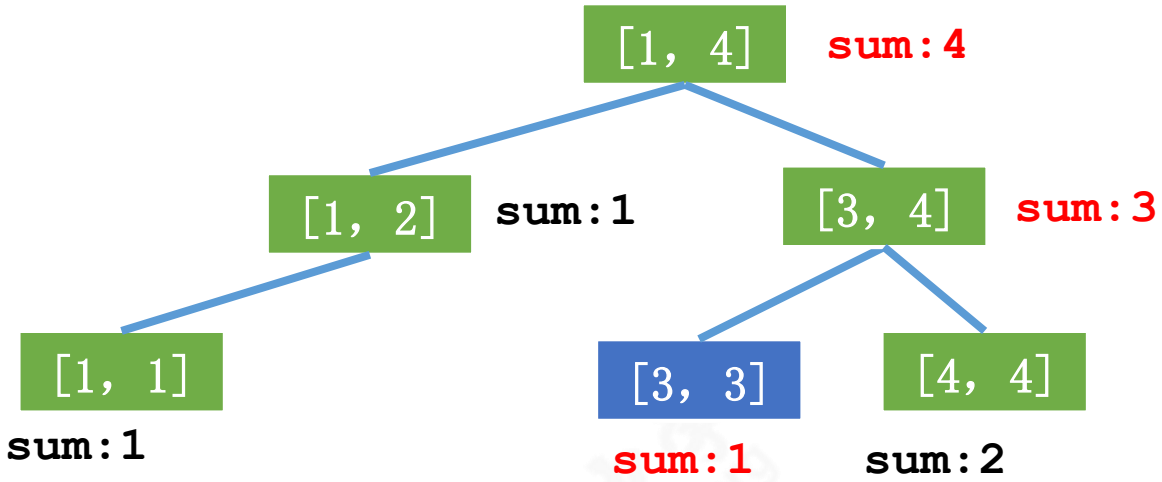
两颗 n 一样的线段树是可以合并的。



线段树的合并

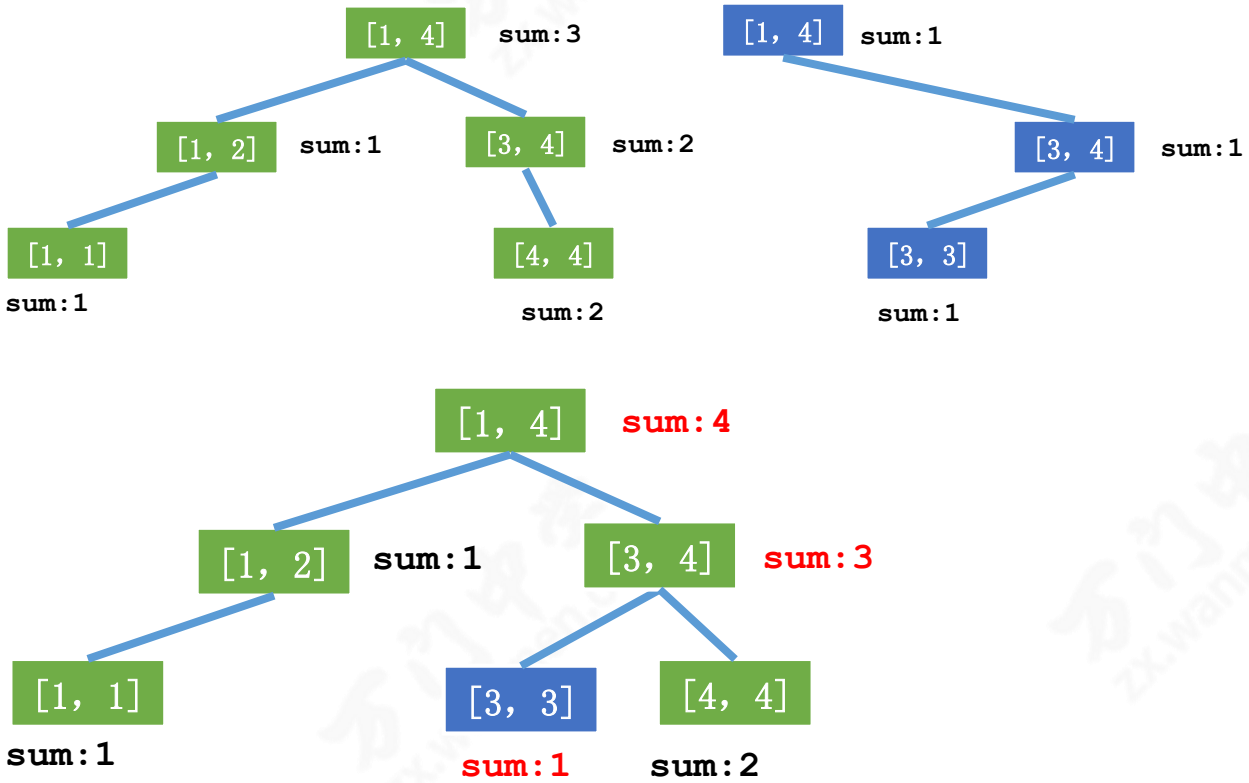
当线段树维护的权值范围［1，n］是一样的时候，线段树的形态是唯一的。

两颗 n 一样的线段树是可以合并的。



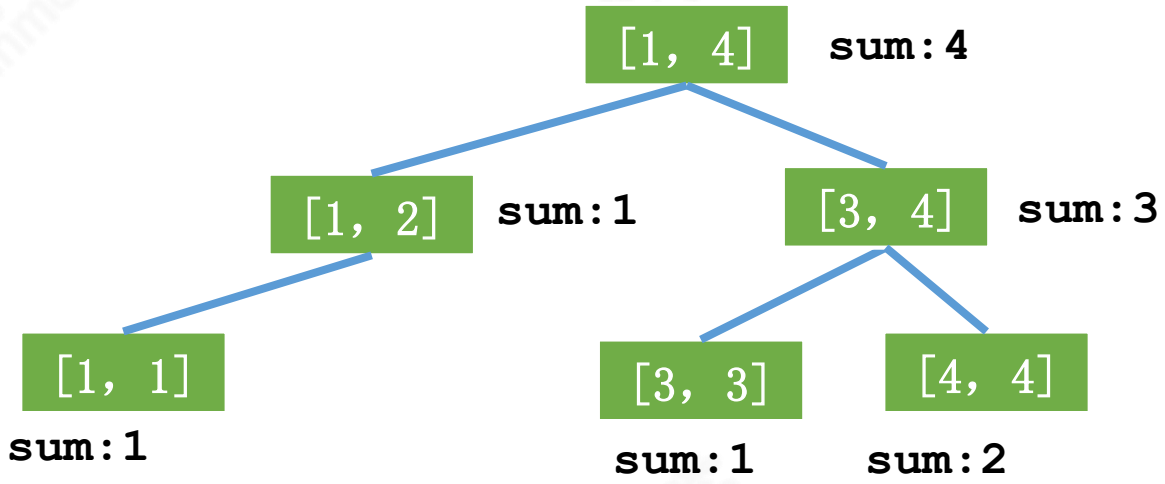
线段树的合并

```
void merge(int &x, int y){
    if (!y) return;
    if (!x){
        x = y;
        return;
    }
    merge(lc[x], lc[y]);
    merge(rc[x], rc[y]);
    update(x);
}
```



线段树的分裂

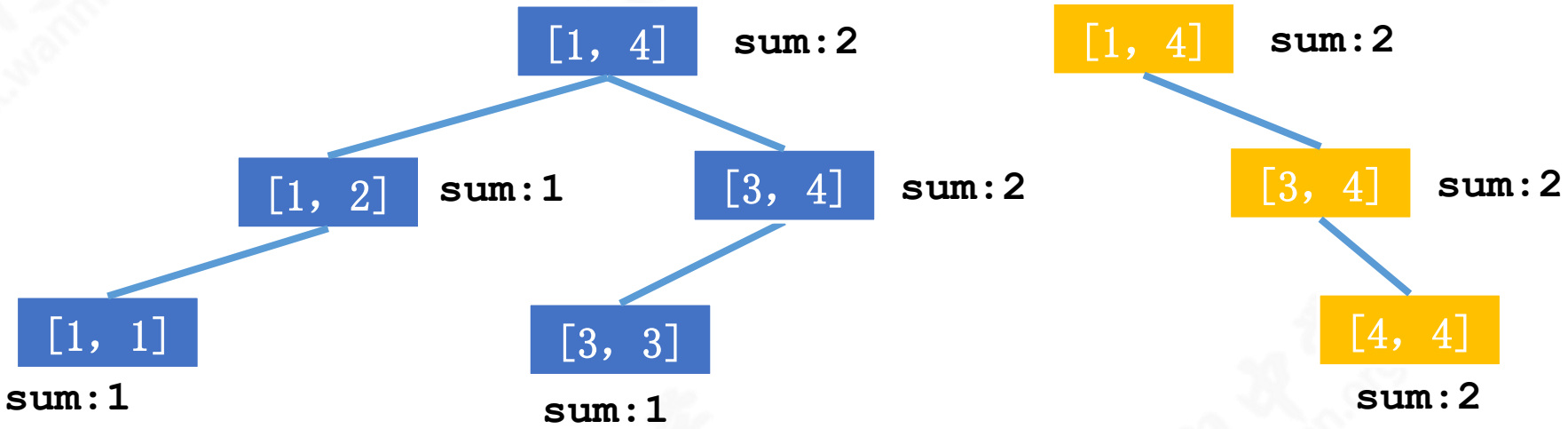
当然一颗线段树也是可以分裂成两颗的，比如我们想把最小的两个节点组成的线段树分裂出来。



线段树的分裂

左边这颗线段树就包含了前 2 小的数。

右边是原来的树减去了左边这颗树剩下的树。



线段树的分裂

```
pair<int,int> split(int x, int k){
      if (!x) return make_pair(0, 0);
      if (k <= sum[lc[x]]) {
            pair<int,int> t = split(lc[x], k);
            ++ tot;
            lc[tot] = t.first;
            lc[x] = t.second;
            sum[tot] = k;
            sum[x] -= k;
            return make_pair(tot, x);
      }
      …
```

线段树的分裂

```
pair<int,int> split(int x, int k){
        …
        else {
                pair<int,int> t = split(rc[x], k - sum[lc[x]]);
                ++ tot;
                rc[tot] = t.second;
                rc[x] = t.first;
                sum[tot] = sum[x] - k;
                sum[x] = k;
                return make_pair(x, tot);
        }
}
```

线段树的合并和分裂

一次合并 / 分裂的时间复杂度可大可小。

但是均摊复杂度是有保证的。

比如 n 个单个节点的线段树合并是 O(n log n)。

一颗 n 个节点的线段树分裂 n 次也是 O(n log n)。

在实际题目中，他们能起到优化时间复杂度的效果。

下节课再见