不能可持久化的分块不是好平衡树

唯有那份眩目　未曾忘却

ɞ 先从一些小问题说起 ɞ

- 给你一个长度为 n 的序列 ma
- m 次询问，每次询问区间 [l, r] 内的值的和
- 我会暴力！

```
ans = 0;
for(int i = l; i <= r; i ++){
    ans += ma[i];
}
```

- 时间复杂度 O(n * m) QAQ
- 我会前缀和预处理！
- $\mathrm{sum}[i] = \sum_{j=1}^{i} ma[j]$
- 每次询问

```
ans = sum[r] - sum[l - 1];
```

- 时间复杂度 O(1) qwq

ɞ 另一个小问题 ɞ

- 给你一个长度为 n 的序列 ma
- m 次操作，每次选择区间 [l, r] 值加 v，最后输出序列
- 我会暴力！

```
for(int i = l; i <= r; i ++){
    ma[i] += v;
}
```

- 时间复杂度 O(n * m) QAQ
- 我会延迟标记！

ɞ 延迟标记大法好 ɞ

- 开辟一个新的 lazy 数组，初始值为 0
- lazy[i] = x 表示 ma 中区间 [l, n] 累计增加了 x
- 对于区间 [l, r] 增加 v

```
lazy[l] += v;
lazy[r + 1] -= v;
```

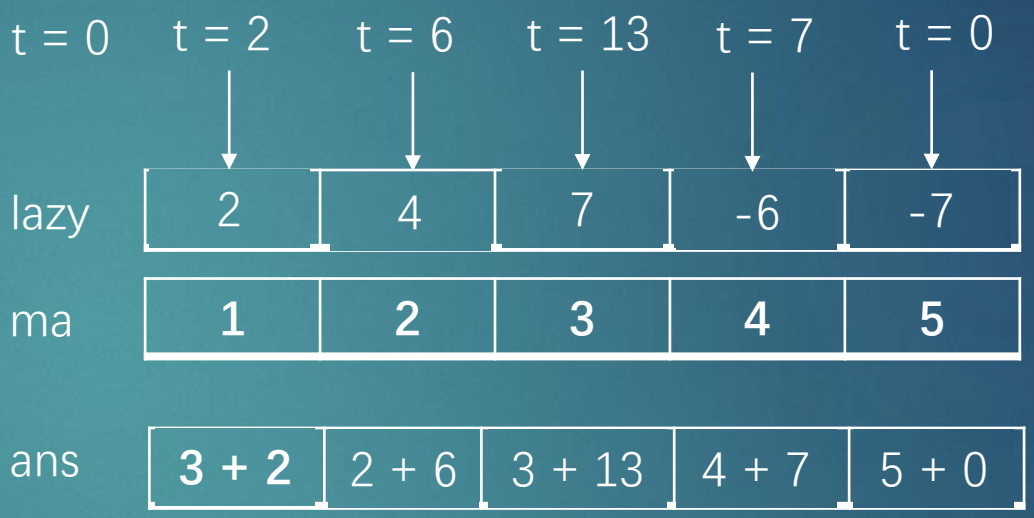
- 最后结果输出

```
t = 0;
for(int i = 0; i < n; i ++){
    t += lazy[i];
    printf("%d ", ma[i] + t);
}
printf("\n");
```

- 单次 O(n) 的修改变成 O(1)！

ɞ 举个栗子 ɞ

- 初始数组 ma 为 1 2 3 4 5
- [1, 3] 加 2
- [3, 4] 加 7
- [2, 3] 加 4
- 最后输出



ɞ 现在问题合并了 ɞ

- 给你一个长度为 n 的序列 ma
- m 次操作
- 每次选择一个区间 [l, r] 使其加 v，或者询问一个区间 [l, r] 的区间和
- 不要再说你会暴力了!
- 内容正式开始qwq

线段树

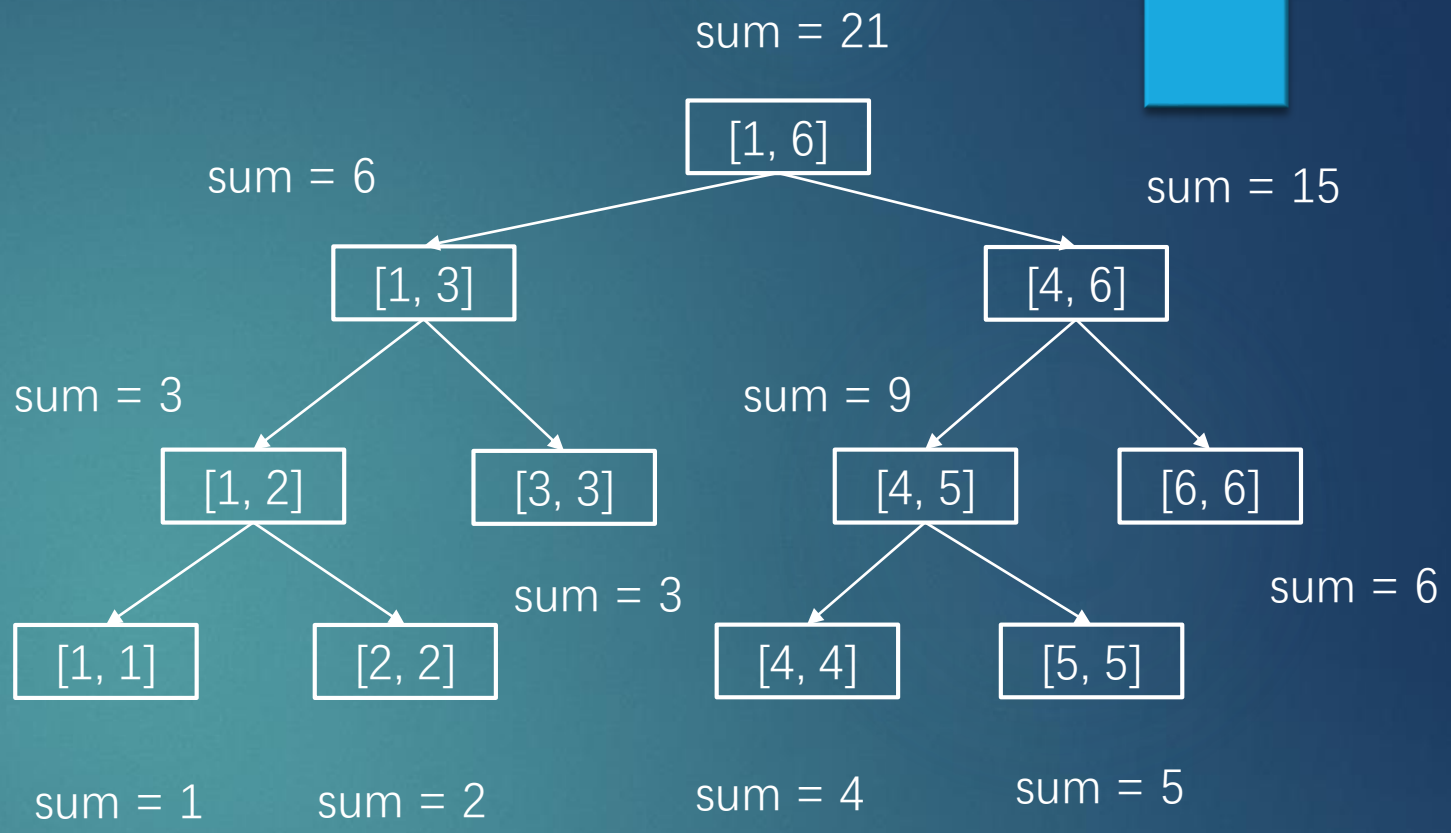
当然关乎美和自信

ɵ 基础定义 ɞ

- 线段树是一种二叉搜索树，即每个结点最多有两棵子树的树结构，左儿子的编号为自身编号 * 2，右儿子编号为自身编号 * 2 + 1
- 通常子树被称作“左子树”（left subtree）和“右子树”（right subtree）
- 线段树的每个结点存储了一个区间
- 所有叶子结点表示的是单位区间(即左右端点相等的区间)，维护对应原始数组下标的值
- 对于表示的区间为 [l, r] 的所有非叶子结点都有左右两棵子树，令 mid = (l + r) / 2 (向下取整)，则它的左儿子表示的区间为[l, mid]，右儿子表示的区间为[mid+1, r]。非叶子节点一定有两个儿子结点，非叶子结点维护他所包含的区间的值

ɞ 来看看线段树 ɞ

- 如图表示的是一个初始数组 ma 为 1 2 3 4 5 的线段树结构
- 树是一个递归结构，两个子结点的区间并正好是父结点的区间，可以通过自底向上的计算更新父节点的值qwq



Build!

- 一个结构体维护每个节点的信息

```
struct Three
{
    int l;
    int r;    /// 当前节点维护区间为 [l, r]
    int sum;  /// [l, r] 内元素区间和
    int lazy; /// 延迟标记
};
```

- 递归建树

```
void Build(int x, int l, int r) /// 表示递归到编号为 x 的节点，该节点维护的区间为 [l, r]
{
    tree[x].l = l;
    tree[x].r = r;
    tree[x].lazy = 0;
    if(l == r){ /// 为叶子节点
        tree[x].sum = ma[l];
    }
    else{
        int mid;

        mid = (l + r) / 2;
        Build(x * 2, l, mid);   ///递归建立左右子树
        Build(x * 2 + 1, mid + 1, r);
        tree[x].sum = tree[x * 2].sum + tree[x * 2 + 1].sum;
        /// 当前区间的区间和为左儿子区间和 + 右儿子区间和
    }
}
```

ɒ Quary! ɓ

得到答案 15 + 3

查询[3, 6]

sum = 21

查询[3, 3]

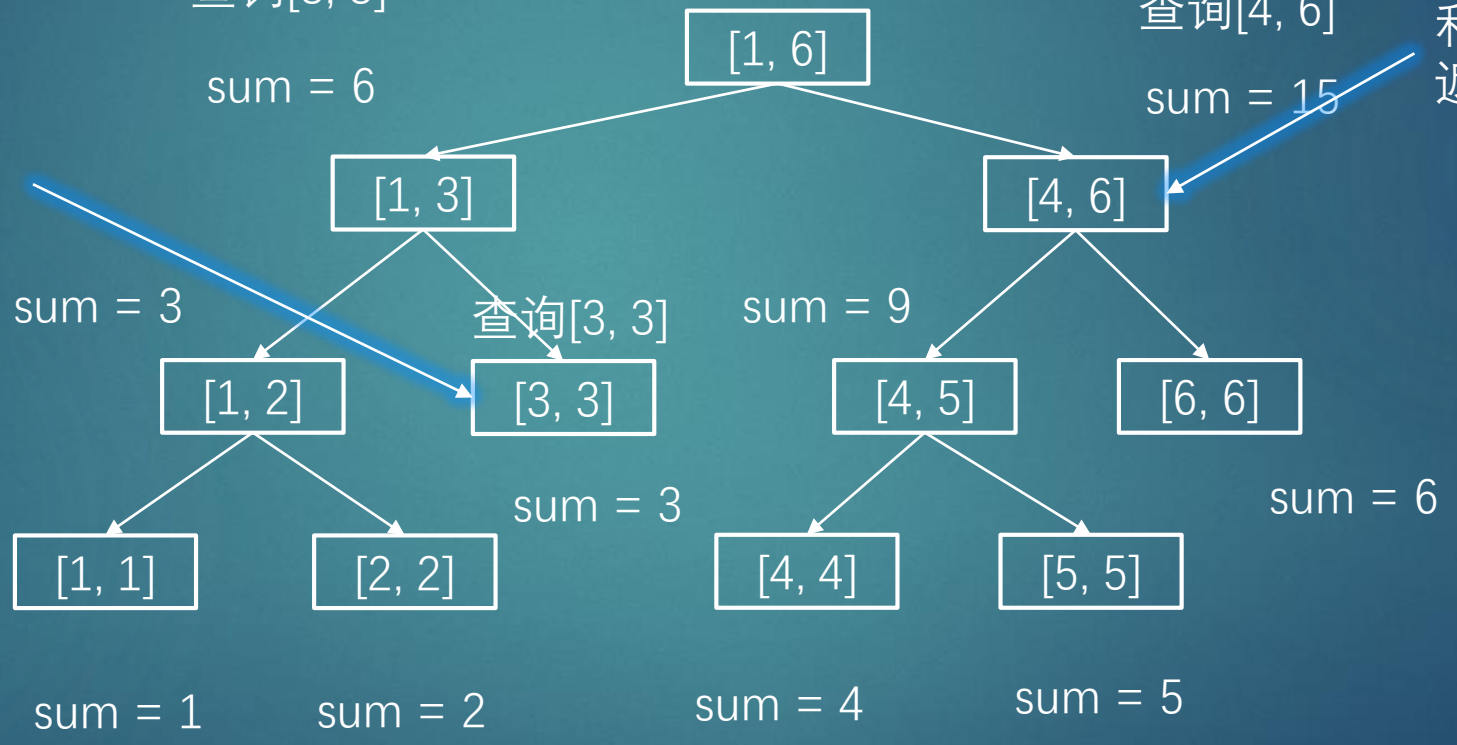
sum = 6

查询[4, 6]

sum = 15

和节点表示区间相等，
返回该节点的值 15!

和节点表示区间相等，
返回该节点的值 3!



剩下的内容咕了

QWQ