二维线段树与四分树

主讲人：邓哲也

二维线段树

对于二维线段树，我们可以理解为线段树套线段树。

比如说我们先对 x 坐标建一颗线段树。

这颗线段树中的每个节点，都对应一个线段树的根节点。

也就是每个节点都存了一颗线段树。

二维线段树

比如 x 和 y 的坐标范围都在 [1, 4]

我们对 (3, 4) 这个位置 + 1.

首先在 x 坐标的线段树上，我们会经过 [1, 4], [3, 4], [3, 3] 这三个节点。

我们只需要在这三个节点对应的线段树上，都给 4 号位置加一即可。

二维线段树

查询矩形和的时候。假设范围是（x1，x2，y1，y2）

先在 x 坐标的线段树上找到［x1，x2］对应的 O(log n) 个节点。

在这些节点对应的 O(log n) 个线段树上查询［y1，y2］的和，相加就是答案。

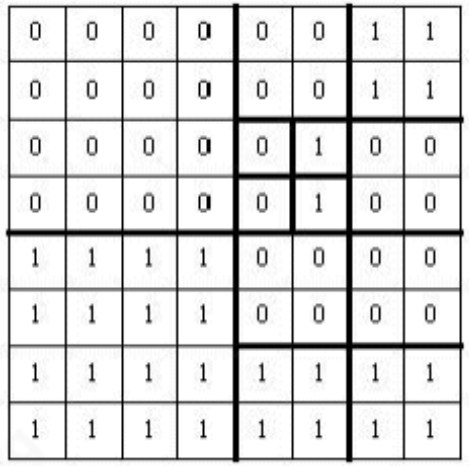
空间复杂度 $O(n^2)$

单次操作复杂度 $O(\log^2 n)$

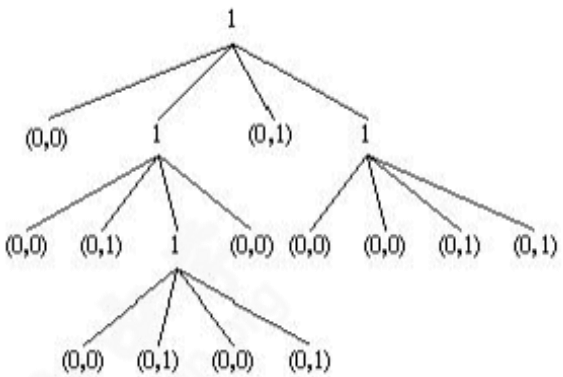
四分树

对于区间，我们从中间切开，分成两个子区间，对应线段树上的子节点。

对于矩形，我们从横、竖两个方向上的中间切开，分成四个子矩形，对应四分数上的 4 个子节点。



(c)



(d)

四分树

我们设 x 的四个子节点的编号为 4x-2，4x-1，4x，4x+1。

```
void update(int x){
      sum[x] = sum[4 * x - 2] + sum[4 * x - 1] + sum[4
* x] + sum[4 * x + 1];
}
```

四分树

```
void build(int xl, int xr, int yl, int yr, int x) {
        if (xl == xr && yl == yr) return;
        int xmid = (xl + xr) >> 1;
        int ymid = (yl + yr) >> 1;
        build(xl, xmid, yl, yr, 4 * x - 2);
        build(xmid + 1, xr, yl, yr, 4 * x - 1);
        build(xl, xr, yl, ymid, 4 * x);
        build(xl, xr, ymid + 1, yr, 4 * x + 1);
        update(x);
}
```

四分树

查询操作类比线段树。

检查查询的矩形是否和四个子节点有交集。

有交集就需要递归到那个子节点下继续查询。

代码比较冗长，这里省略。

下节课再见