

# 解决动态统计问题的两把利刃




## ——剖析线段树与矩形切割

广东北江中学

薛矛



# 目录

- 1 线段树
  - 2 矩形切割
  - 3 线段树与矩形切割的比较
  - 4 总结
- 



# 1 线段树

## ※ 1.1 线段树的结构

线段树是一棵二叉树，线段 $[a,b]$ 的左右儿子分别为 $[a, \lfloor (a+b)/2 \rfloor]$ 和 $[\lfloor (a+b)/2 \rfloor, b]$ 。线段树的叶子结点是长度为1的单位线段 $[a, a+1]$ 。

## ※ 1.2 线段树的建立

**Procedure** MakeTree(a,b)

**Var** Now:Longint

**Begin**

tot  $\leftarrow$  tot + 1      Now  $\leftarrow$  tot

Tree[Now].a  $\leftarrow$  a      Tree[Now].b  $\leftarrow$  b

**If** a + 1 < b **then**

    Tree[Now].Left  $\leftarrow$  tot + 1      MakeTree(a,  $\lfloor (a+b)/2 \rfloor$  )

    Tree[Now].Right  $\leftarrow$  tot + 1      MakeTree( $\lfloor (a+b)/2 \rfloor$  , b)

**End**



# 1 线段树

## ※ 1.3 线段的插入和删除

可增加一个Cover域来计算一条线段被覆盖的次数:

**Type** TreeNode=**Record**

a,b,Left,Right,Cover:Longint

**End**

**插入一条线段[c,d]**

**Procedure** Insert(Num)

**Begin**

**If** (c<Tree[Num].a)and(Tree[Num].b<d) **then**

Tree[Num].Cover ← Tree[Num].Cover + 1

**Else**

**If**  $c < \lfloor (Tree[Num].a + Tree[Num].b) / 2 \rfloor$  **then**

Insert(Tree[Num].Left)

**If**  $d > \lfloor (Tree[Num].a + Tree[Num].b) / 2 \rfloor$  **then**

Insert(Tree[Num].Right)

**End**



# 1 线段树

删除一条线段[c,d]

**Procedure** Delete(Num)

**Begin**

**If** ( $c < \text{Tree}[\text{Num}].a$ ) and ( $\text{Tree}[\text{Num}].b < d$ ) **then**

$\text{Tree}[\text{Num}].\text{Cover} \leftarrow \text{Tree}[\text{Num}].\text{Cover} - 1$

**Else**

**If**  $c < \lfloor (\text{Tree}[\text{Num}].a + \text{Tree}[\text{Num}].b) / 2 \rfloor$  **then**

        Delete( $\text{Tree}[\text{Num}].\text{Left}$ )

**If**  $d > \lfloor (\text{Tree}[\text{Num}].a + \text{Tree}[\text{Num}].b) / 2 \rfloor$  **then**

        Delete( $\text{Tree}[\text{Num}].\text{Right}$ )

**End**





# 1 线段树

## ※1.4 线段树的简单应用

线段树能解决一些最基本的统计问题。但是如果处理一些需要进行修改的动态统计问题，困难就出现了。

**例1** 在数轴上进行一系列操作。每次操作有两种类型，一种是在线段 $[a,b]$ 上涂上颜色，另一种将 $[a,b]$ 上的颜色擦去。问经过一系列的操作后，有多少条单位线段 $[k,k+1]$ 被涂上了颜色。





# 1 线段树

线段树中线段的删除只能把已经放入的线段删掉。而在这道题目中，若给线段 $[1,15]$ 涂了颜色，可以把 $[4,9]$ 上的颜色擦去。但线段树中只是插入了 $[1,15]$ 这条线段，要删除 $[4,9]$ 这条线段显然是做不到的。因此，我们有必要对线段树进行改进。



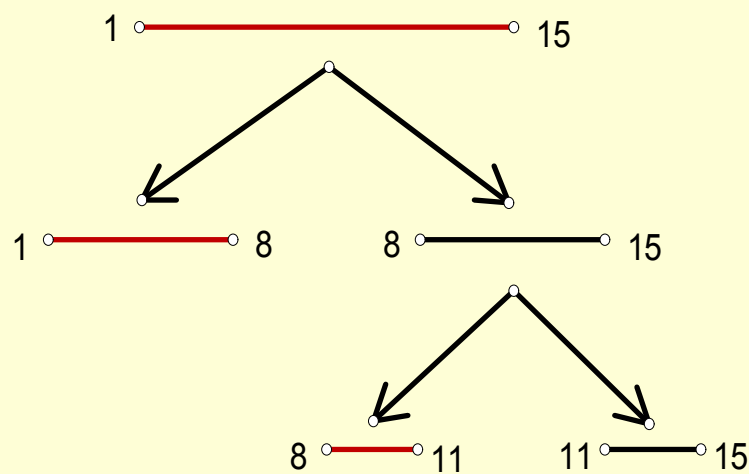
# 1 线段树

## ※ 1.5 线段树的改进

不难想到把 $[1,15]$ 这条线段删去，再插入线段 $[1,4]$ 和 $[9,15]$ 。但事实上并非如此简单。如下图：

若先前已插入了线段 $[1,8]$ ， $[8,11]$ 。按上面的做法，只把 $[1,15]$ 删去，然后插入 $[1,4]$ ， $[9,15]$ 的话， $[1,8]$ ， $[8,11]$ 这两条线段并没有删去，很明显是与实际不符的。于是 $[1,8]$ ， $[8,11]$ 也要修改。

但若出现以下这种情况：








# 1 线段树

以线段 $[1,15]$ 为根的整棵线段树中的所有结点之前都已经插入过，即我们曾经这样涂过颜色：

$[1,2], [2,3], \dots, [14,15], [1,3], [3,5], \dots,$


$[13,15], [1,5], \dots, [1,15]$ 。然后把 $[1,15]$ 上的颜色擦去。那么整个线段树中的所有结点的状态就都与实际不符了，全都需要修改。修改的复杂度就是线段树的结点数。线段稍长复杂度就很高了。





# 1 线段树

为了解决这个问题，我们为每个结点增加一个标记域 **$bj$** 。

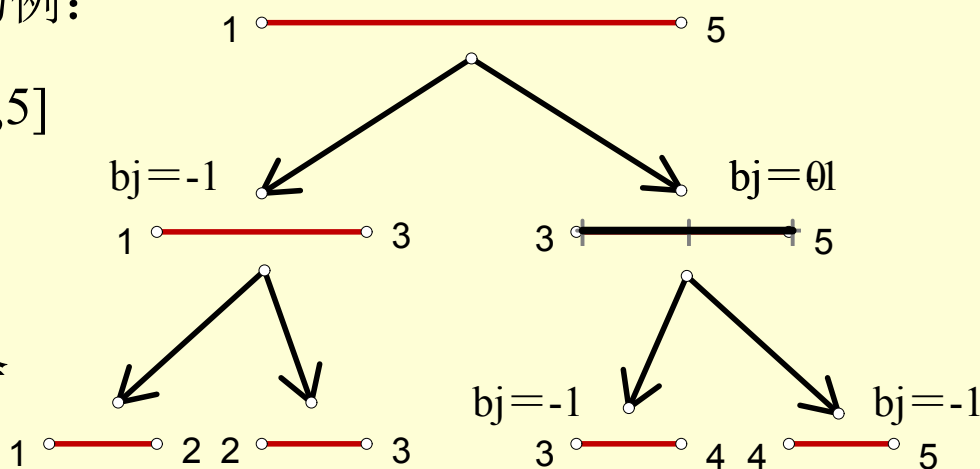
- 1、在擦去线段 $[a,b]$ 之后，给它的左儿子和右儿子都做上标记，令它们的 $bj=-1$ 。而不需要对整棵树进行修改。
  - 2、以后每次访问某条线段，首先检查它是否被标记，若被标记，则进行如下操作：
    - ① 将该线段的状态改为未被覆盖，并把该线段设为未被标记， $bj=0$ 。
    - ② 把该线段的左右儿子都设为被标记， $bj=-1$ 。
- 

# 1 线段树

这样做的原理很简单，以右图为例：

把线段 $[1,5]$ 擦去后，给 $[1,3]$ ， $[3,5]$ 加上标记。

若以后我们需要用到线段 $[3,4]$ ，就必须先访问 $[3,5]$ ，因为 $[3,5]$ 被标记，我们访问它之后标记就会传递给 $[3,4]$ 和 $[4,5]$ 。



$[3,4]$ 就给标记上了。也就是说，标记会顺着访问 $[3,4]$ 的路径一直传递下去。

所以当我们需要用到下面的某条线段时，标记就会传到它那里去，使它得到更新，避免错误的发生。而对于那些以后用不到的线段，就没有更新的必要了，因此我们也不会访问到它和更新它，这样就避免了无用功的产生，提高了程序效率。



# 1 线段树

进行标记更新的代码如下：

**Procedure** Clear(Num)

**Begin**

Tree[Num].Cover  $\leftarrow$  0


Tree[Num].bj  $\leftarrow$  0

Tree[Tree[Num].Left].bj  $\leftarrow$  -1

Tree[Tree[Num].Right].bj  $\leftarrow$  -1

**End**

在访问编号为Num的线段前判断后调用





# 1 线段树

引入标记域后例1就能顺利解决了，做法大体上是一样的，具体的细节可以参考论文，这里就不多说了。






# 1 线段树

## 小结:

如果我们对整条线段 $[a,b]$ 进行操作的话, 我们就可以只是给 $[a,b]$ 的左右儿子做上标记, 而无需对以 $[a,b]$ 为根的整棵子树中的所有结点进行修改。





# 1 线段树

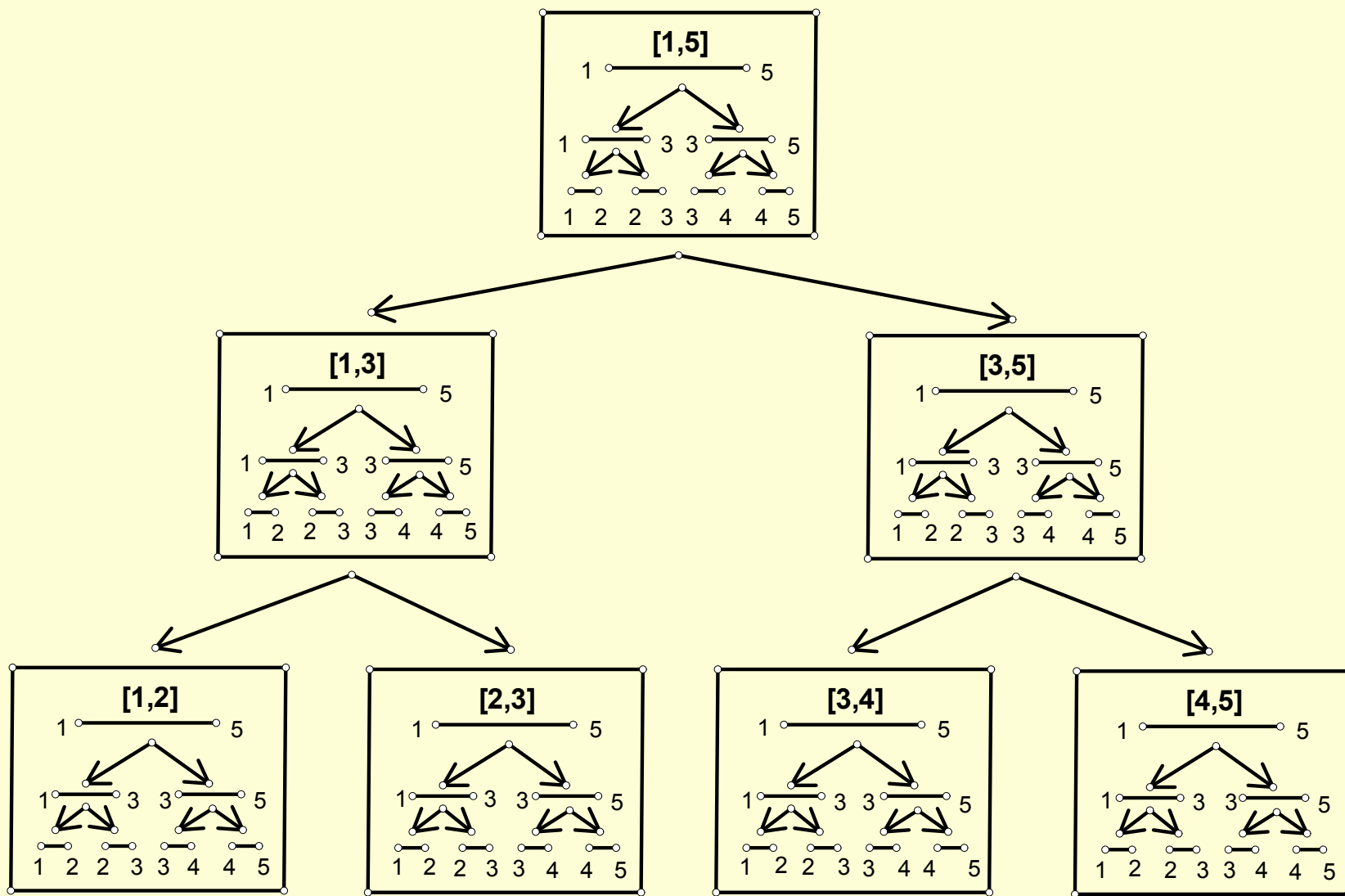
## ※ 1.6 线段树的推广

线段树处理的是线性统计问题，而我们往往会遇到一些平面统计问题和空间统计问题，因此我们需要推广线段树，使它变成二维线段树和 multidimensional 线段树。

将一维线段树改成二维线段树，有两种方法。一种就是给原来线段树中的每个结点都加多一棵线段树，即“树中有树”。如下图：



# 1 线段树








# 1 线段树

容易算出，用这种方法构造一棵矩形 $(x1,y1,x2,y2)$ 的线段树的空间复杂度为  $O(\text{Long\_x} \times \text{Long\_y})$ 。其中  $\text{Long\_x}$ ， $\text{Long\_y}$ 分别表示矩形的长和宽。

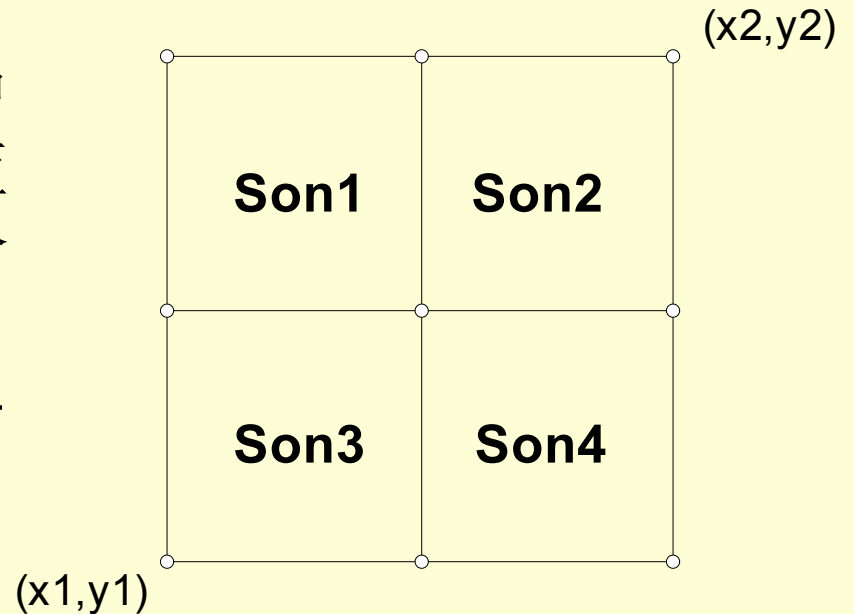
相应地，时间复杂度为  $O(n \times \log_2(\text{Long\_x}) \times \log_2(\text{Long\_y}))$ 。其中 $n$ 为操作数。

由于这种线段树有两层，处理起来较麻烦。



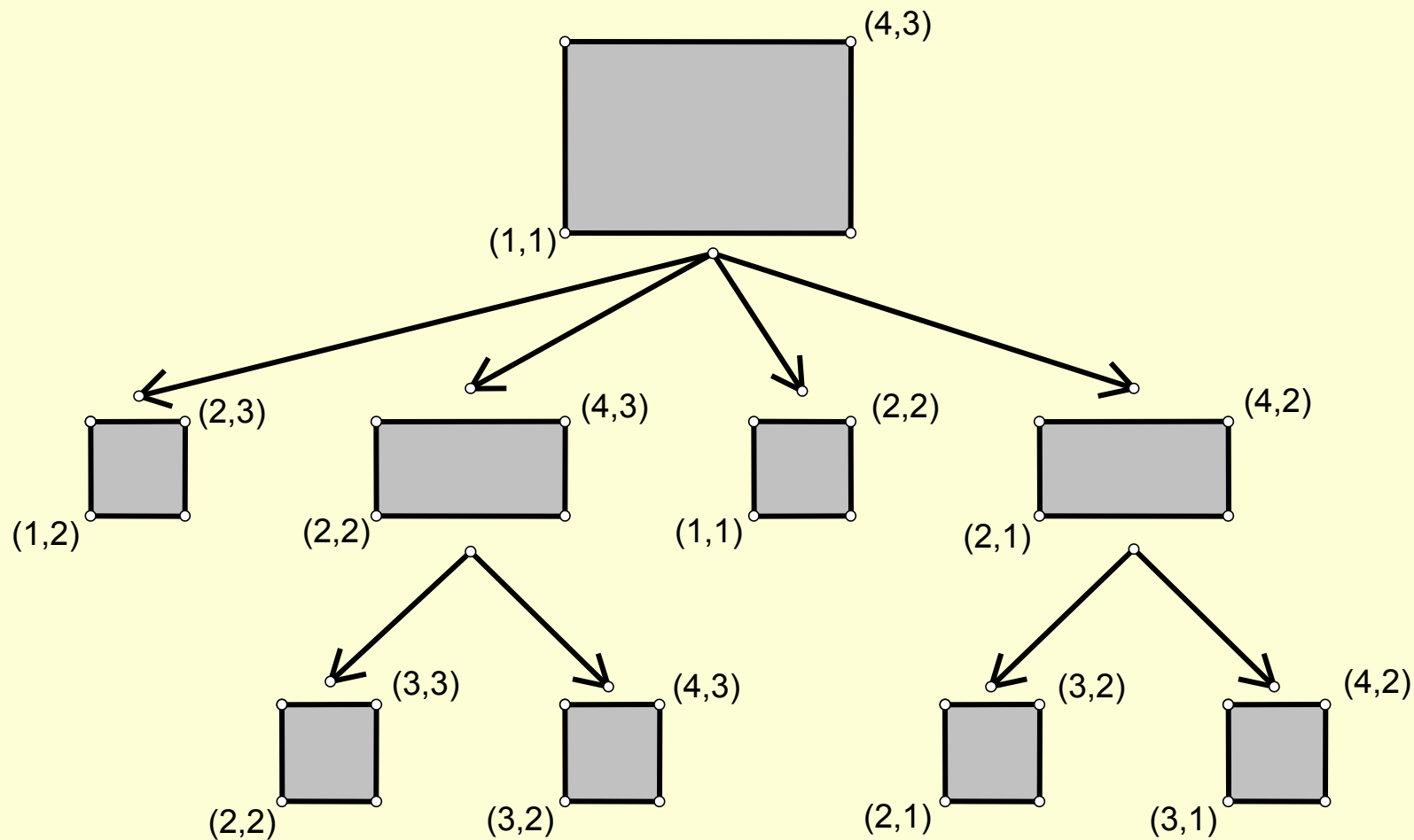
# 1 线段树

另一种方法是将线段树结点中的线段变成矩形，从而变为矩形树。因此矩形树用的是四分的思想，每个矩形分割为4个子矩形。矩形 $(x1,y1,x2,y2)$ 的4个儿子如右图所示



# 1 线段树

这是一棵以矩形(1,1,4,3)为根的矩形树：






# 1 线段树

以 $(x1, y1, x2, y2)$ 为根的矩形树的空间复杂度也是  $O(\text{Long\_x} \times \text{Long\_y})$ 。

由于它只有一层，处理起来比第一种方法方便。而且在这种矩形树中，标记思想依然适用。而第一种方法中，标号思想在主线段树上并不适用，只能在第二层线段树上使用。


但是这种方法的时间复杂度可能会达到  $O(n \times \text{Long\_x})$ 。比起第一种来就差了不少。





# 1 线段树

对于多维的问题，第一种方法几乎不可能使用。因此我们可以仿照第二种方法。例如对于 $n$ 维的问题。我们构造以 $(a_1, a_2, a_3, \dots, a_n, b_1, b_2, b_3, \dots, b_n)$ 为根的线段树，其中 $(a_1, a_2, a_3, \dots, a_n)$ 表示的是左下角的坐标， $(b_1, b_2, b_3, \dots, b_n)$ 表示的是右上角的坐标。用的是 $2^n$ 分的思想，构造出一棵 $2^n$ 叉树。结点的个数变为 $2^n \times (b_1 - a_1) \times (b_2 - a_2) \times \dots \times (b_n - a_n)$ 。





# 1 线段树

## ※ 1.7 线段树小结

线段树在改进和推广之后，做到了高效地解决更多的问题。因其适用范围广和实现上的方便，线段树不失为一个优秀的方法。但线段树还是有一些缺陷的，下文将在与矩形切割进行比较的时候提及。

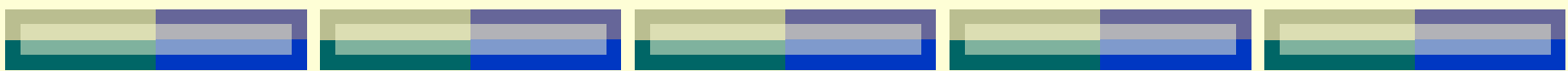




## 2 矩形切割

矩形切割是一种处理平面上矩形的统计的方法。许多统计类的问题通过数学建模后都能转化为用矩形切割来解决。矩形切割的原型是线段切割。我们先来看看线段切割的思想。





## 2.1 线段切割

例2 在数轴上进行一系列操作。每次在线段 $[a,b]$ 上涂色，涂的颜色可以有多种，同一线段上后涂的颜色会覆盖先涂的颜色。经过一系列操作后，对每一种颜色都求出含有该种颜色的单位线段 $[k,k+1]$ 的条数。







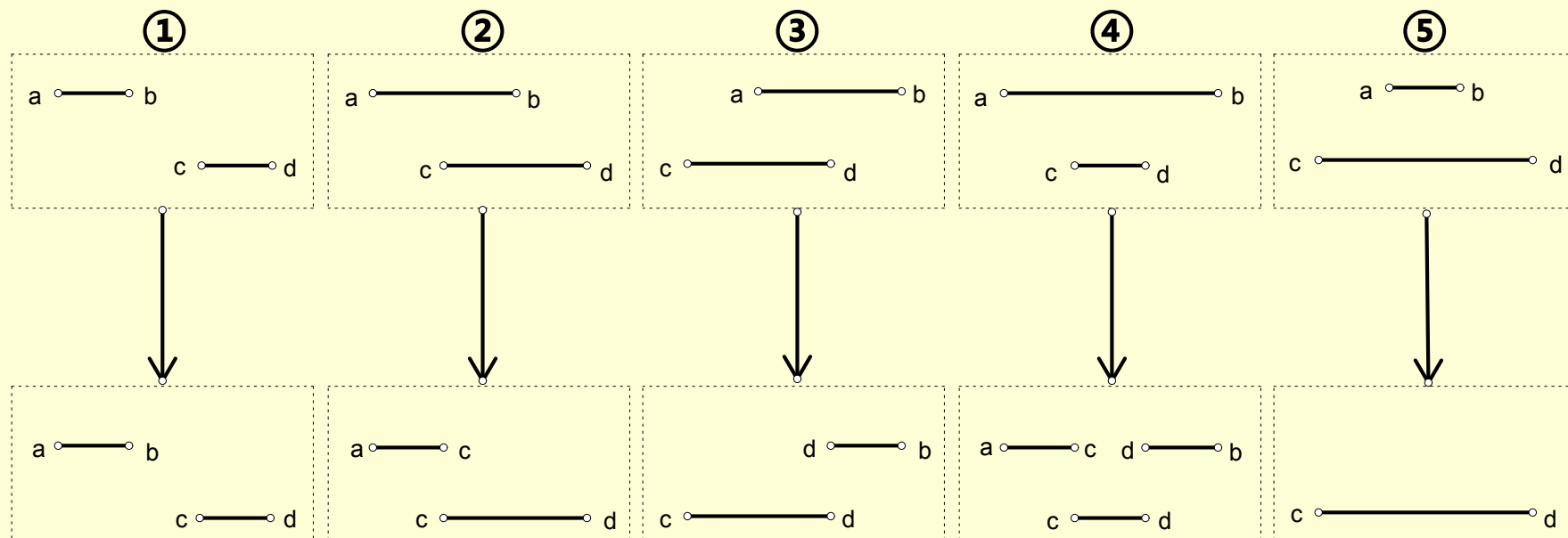
## 2.1 线段切割

由于线段之间会出现重叠，我们引入线段切割的方法对集合中的线段进行动态维护，使得所有线段两两不重叠。那么最后只需直接将同种颜色的线段长度累加，就能得出答案。

若线段集合中本来有一根线段 $[a,b]$ ，现在加入一根新线段 $[c,d]$ 。那么它们之间的位置关系可能有以下几种：



## 2.1 线段切割



对于每一种位置关系，我们都可以通过切割线段 $[a, b]$ ，并删除某些小段，使得它与新线段 $[c, d]$ 不重叠。



## 2.1 线段切割

判断线段 $[a,b]$ ,  $[c,d]$ 是否重叠的方法

若 $a \geq d$ 或者 $c \geq b$ , 就不重叠, 否则重叠。

切割线段 $[a,b]$ 的方法

取线段 $[a,b]$ ,  $[c,d]$ 的交集 $[k1,k2]$ 。

若 $a < k1$ , 则加入线段 $[a,k1]$ ;

若 $k2 < b$ , 则加入线段 $[k2,b]$ 。

删除线段 $[a,b]$

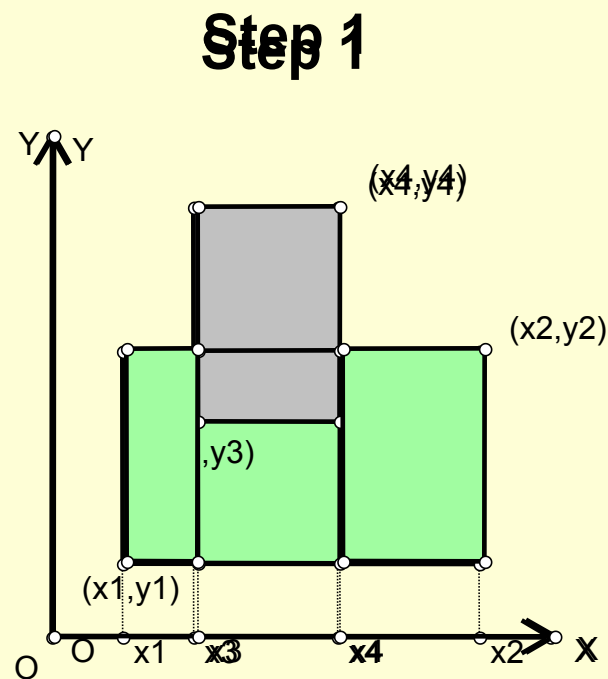


## 2 矩形切割

类似地，我们可以将矩形切割正交分解，先进行x方向上的切割，再进行y方向的切割。

如右图，现在加入矩形 $(x_3, y_3, x_4, y_4)$ ，对矩形 $(x_1, y_1, x_2, y_2)$ 进行切割。

Step 1: 首先从x方向上切。把线段 $(x_1, x_2)$ 切成 $(x_1, x_3)$ ， $(x_4, x_2)$ 两条线段。于是切出两个矩形—— $(x_1, y_1, x_3, y_2)$ ， $(x_4, y_1, x_2, y_2)$ 。把它们加入到矩形集合中

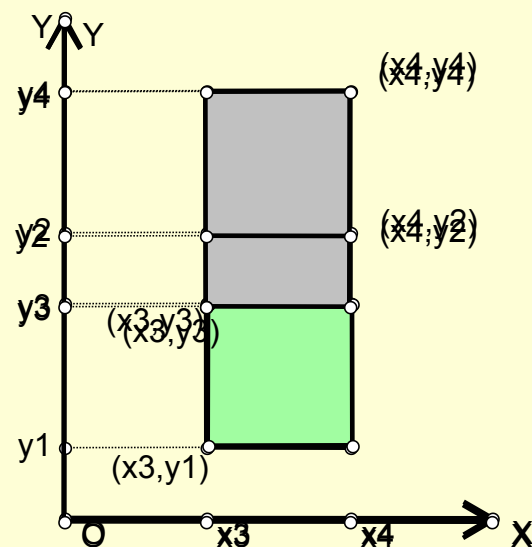


## 2 矩形切割

Step 2:接着我们再进行y方向上的切割。把线段 $(y1,y2)$ 切成 $(y1,y3)$ 。相应地又得到一个矩形 $(x3,y1,x4,y2)$ 。把它加入到矩形集合中。

Step 3:把原来的矩形 $(x1,y1,x2,y2)$ 从矩形集合中删去。


### Step 2





## 2 矩形切割

我们可以归纳出**矩形切割的思想**：

- 1、先对被切割矩形进行x方向上的切割。取 $(x1, x2)$ ,  $(x3, x4)$ 的交集 $(k1, k2)$ 。
    - ① 若 $x1 < k1$ ，则加入矩形 $(x1, y1, k1, y2)$
    - ② 若 $k2 < x2$ ，则加入矩形 $(k2, y1, x2, y2)$
  - 2、再对切剩的矩形 $(k1, y1, k2, y2)$  进行y方向上的切割。取 $(y1, y2)$ ,  $(y3, y4)$ 的交集 $(k3, k4)$ 
    - ① 若 $y1 < k3$ ，则加入矩形 $(k1, y1, k2, k3)$
    - ② 若 $k4 < y2$ ，则加入矩形 $(k1, k4, k2, y2)$
  - 3、把矩形 $(x1, y1, x2, y2)$ 从矩形集合中删除。
- 




## 2 矩形切割

### ※ 2.3 矩形切割的推广

本着矩形切割的思想，我们可以把它推广到 $n$ 维。

两个 $n$ 维物体有重叠部分的充要条件就是它们在 $n$ 个方向上都存在交集。

切割的方法也是类似的：先在 $x$ 方向上切，然后在 $y$ 方向上切，接着在 $z$ 方向上切，……，一直到在第 $n$ 个方向上切。





## 2 矩形切割

### ※ 2.4 矩形切割的应用

矩形切割可以解决几何类的统计问题。而对于其它的统计类问题，只要能建立起矩形切割的数学模型，也能用它来解决。具体例子请参考论文。








### 3 线段树与矩形切割的比较

对两种方法进行比较，我们可以先从复杂度入手

#### ※ 3.1 线段树的时空复杂度

- 线段树的空间复杂度是 $O(\text{Long\_x})$ 。
  - 二维线段树是 $O(\text{Long\_x} * \text{Long\_y})$ 。
  - 三维线段树是 $O(\text{Long\_x} * \text{Long\_y} * \text{Long\_z})$ 。
  
  - 线段树的时间复杂度为  $O(n * \log_2(\text{Long\_x}))$ 。
  - 矩形树是 $O(n * \log_2(\text{Long\_x}) * \log_2(\text{Long\_y}))$ 。
  - 方块树是 $O(n * \log_2(\text{Long\_x}) * \log_2(\text{Long\_y}) * \log_2(\text{Long\_z}))$ 。
- 




### 3 线段树与矩形切割的比较

#### ※ 3.2 矩形切割的时空复杂度

矩形切割的时空复杂度较难估算。因为放入的矩形不同，切割出来的矩形数目也就不同。

矩形切割的**空间复杂度**是由矩形集合中矩形数目的峰值 $n$ （即曾经在矩形集合中出现的矩形数目的最大值）决定的。我们先做一些数据随机生成 $m$ 个矩形，看看峰值 $n$ 会是多少。如下表：



### 3 线段树与矩形切割的比较

表1

矩形数m	100	500	1000	5000	10000	50000	100000
峰值n	239	479	680	1015	1296	1741	2092

这些数据中的矩形是随机生成的。其中m是输入数据中的矩形个数。对于数据中的每个矩形 $(x1,y1,x2,y2)$ 都有： $0 \leq x1 < x2 \leq 60000$ ， $0 \leq y1 < y2 \leq 60000$ 。其中对矩形集合中矩形数目的峰值n，计算方法是：对同一个m值，生成10组数据，得出10个n值。取这些结果的平均值作为n的值。

我们发现，随着矩形数m增大的加剧，峰值n只是维持在较低的水平。因此空间复杂度十分低。但对于一些特意构造的极端数据，空间复杂度达到了 $O(m^2)$ 。（论文中有详细介绍）



### 3 线段树与矩形切割的比较

矩形切割的**时间复杂度**为 $O(n*m)$ 。即矩形数 $m$ 乘以峰值 $n$ 。由表1可以看到矩形切割的时间效率还是挺高的。而对于极端数据，时间复杂度达到了 $O(m^3)$ ，效率就很低了。






## 3 线段树与矩形切割的比较

### ※ 3.3 线段树与矩形切割适用范围的比较

根据线段树和矩形切割的复杂度。我们就可以思考出它们的适用范围。

线段树的空间复杂度是固定的，若线段的端点取值范围很大，线段树的空间复杂度将会十分大甚至无法承受。特别是在矩形树和方块树中。而矩形切割在这方面就十分有优势了。它是用变量来存边界的，不受端点取值范围的影响。

线段树的时间复杂度很小，只有 $O(n\log_2 n)$ ，因此对于操作数较多的题目十分适用。然而对矩形切割来说操作数一多，效率就不高了。





### 3 线段树与矩形切割的比较

#### ※ 3.3 线段树与矩形切割适用范围的比较

结论：

对边界范围小，操作数多的题目，我们选择线段树；对边界范围大，操作数少的题目，我们选择矩形切割。





## 4 总结

- 1、发现和提出问题，从多方面思考，研究解决方案，进行改进与推广。
- 2、善于比较各种方法，分析优缺点，总结其适用条件。





谢谢大家

