线段树的应用

广西柳铁一中 林涛

【摘要】

在竞赛解题中，常遇到与区间有关的操作，比如统计若干矩形并的面积，记录一个区间的最值、总量，并在区间的插入、删除和修改中维护这些最值、总量。

线段树拥有良好的树形二分结构，能够高效的完成这些操作，本文将介绍线段树的各种操作以及一些推广。

本文通过 3 个例子：《蛇》 、《空心长方体》、 《战场统计系统》，讲述线段树中基本的插入、删除、查找操作，和不规则的修改和删除操作，以及到二维的推广。

关键字：线段树　　二分　　子树收缩　叶子释放　面积树

【正文】

1. 线段树的定义及特征

定义 1：线段树

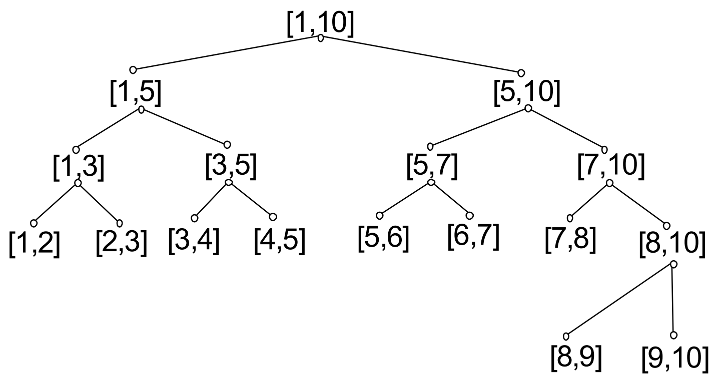
一棵二叉树，记为 T (*a*,*b*)，参数 *a*,*b* 表示该节点表示区间[*a*,*b*]。区间的长度 *b*-*a* 记为 *L*。递归定义 T[*a*,*b*]：

若 *L*>1 ：[*a*, (*a*+*b*) div 2]为 T 的左儿子

　　　　[(*a*+*b*) div 2,*b*]为 T 的右儿子。

若 *L*=1 ：T 为一个叶子节点。

表示区间[1, 10]的线段树表示如下：



(以下取对数后均向上取整)

定理 1：线段树把区间上的任意一条线段都分成不超过 2log*L* 条线段

证明：(1)在区间(*a*，*b*)中，对于线段(*c*，*d*)，如果(*c*<=*a*) 或 (*d*>=*b*)，那么线段在(*a*，*b*)中被分为不超过 log(*b*-*a*)。

用归纳法证明，如果是单位区间，最多被分为一段，成立。

如果区间(*a*，*b*)的左儿子与右儿子成立，那么如果当 *c*<=*a* 时，

1. 若 *d*<=(*a*+*b*)div2 那么相当与其左儿子分该线段，所分该线段数树不超过 log((*a*+*b*)div 2-*a*)，即不超过 log(*b*-*a*),成立。

2．若 $d>(a+b)$ div 2 那么相当于该线段被分为它左儿子表示的线段，加上右儿子分该线段，线段数不超过 $1+\log(b-(a+b)$ div $2)$,也不超过 $\log(b-a)$，成立。

对于 $d>=b$ 的情况证明类似，不再赘述。

(2)在区间(a,b)中，对于任意线段也用归纳法证明。

对于单位区间，最多分为一段，成立。

若$(a，b)$的左儿子与右儿子均成立，则对于线段$(c，\mathrm{d})$

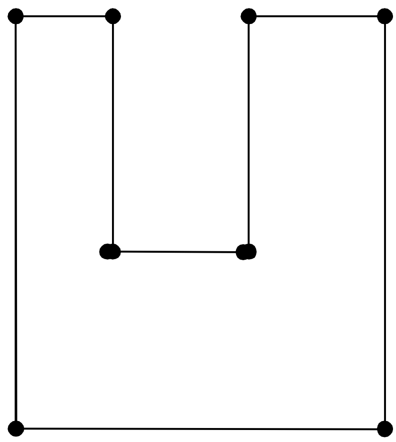
1．若 $\mathrm{d}<=(a+b)$div 2 则该区间所分该线段等于其左儿子区间所分该线段，线段数小于 $\log((a+b)$ div $2-a)<2\log(b-a)$，成立。

2．若 $c>(a+b)$ div 2 则该区间所分该线段等于其右儿子区间所分该线段，线段数小于 $\log(b-(a+b)$ div $2)<2\log(b-a)$，成立。

3．若 1、2 均不成立，则此线段在左儿子区间分该线段满足 $d>$V.Lson.b，分该线段数不超过 $\log(b-(a+b)$ div $2)$，而在右儿子区间分该线段满足 $c<=$V.Rson.a,分该线段不超过 $\log((a+b)$ div $2-1)$，所以在该区间分该线段不超过 $2\log(b-a)$，成立。

这个结论为线段树能在 $O(\log L)$的时间内完成一条线段的插入、删除、查找等工作，提供了理论依据。

【例题一】蛇[1]



在平面上有 N 个点，现在要求一些线段，使其满足以下要求：

a． 这些线段必须闭合

b． 线段的端点只能是这 N 个点

c． 交于一点的两条线段成 90 度角

d． 线段都必须平行于坐标轴

e． 所有线段除在这 N 个点外不自交

f． 所有线段的长度之和必须最短

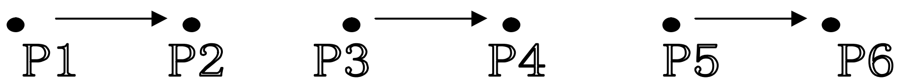
如果存在这样的线段，则输出最小长度，否则输出 0。

【问题分析】

从该题的要求入手，先构出符合要求的图，再解决线段长度之和最小的问题。

1．题目显然要求一个以给定的 N 个点为顶点的 N 多边形。所有线段都要和坐标轴平行，所以每个点只能与上下左右四个点相连。由于与一个点相连的两条线段成 90 度，每个顶点必须与一条平行于 X 轴和一条平行于 Y 轴的线段相连。

2．将所有点排序后发现，在同一水平线上的点中，设这些点为 P1，P2，P3，P4……Pn，P1 要有一条平行于 X 轴的线段与其相连，就必须连它右边的点——P2，而 P3 如果再连 P2，P2 就有两条平行于 X 轴的线段和它相连，所以 P3 只能连 P4，P5 只能连 P6……，同一垂直线上的点也是如此，所以线段的构造是唯一的，那么最小长度的问题就解决了。



3．由于解是唯一的，而是否相连只要广度扩展就可以判断了，所以关键在于判断由上述方法所构出线段是否合法——满足线段不在 N 个点之外自交：

[1] Saratov State University Problem Archive, 1028, Snake. 本题考查了基本的插入、删除和查找

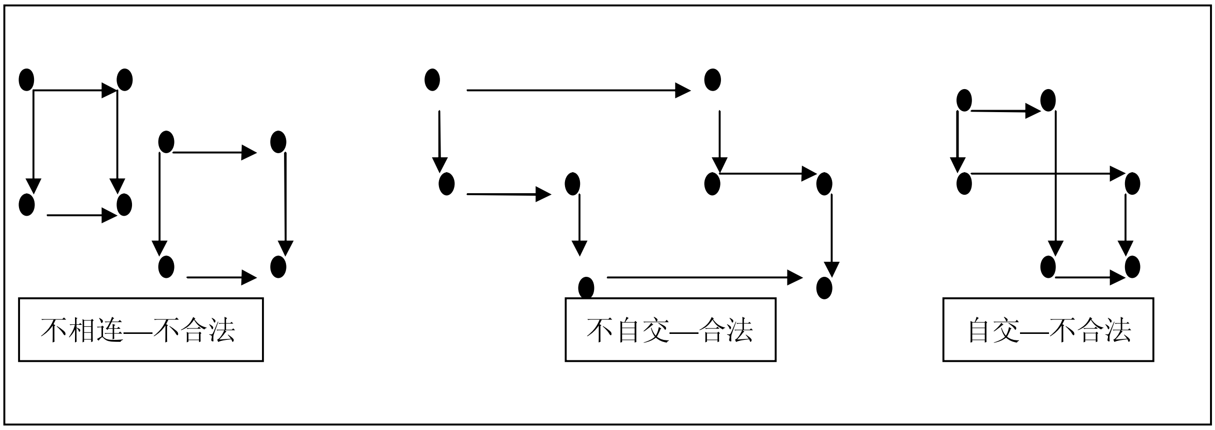
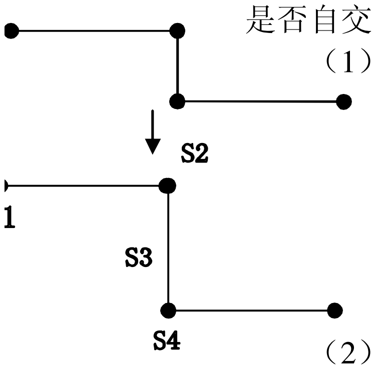
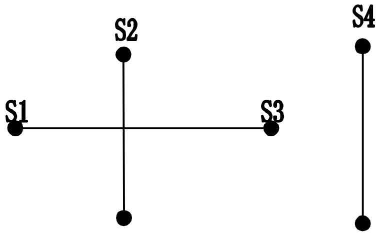


图 1. 合法性判断

4．由于所有线段与坐标轴之一平行，有明显的区间性，可以想到用线段树判断是否自交：





（1）　由于只可能是与 X 轴平行的线段和与 Y 轴的线段相交，所以可以只考虑与 Y 轴平行的线段是否有线段与之相交。如果线段 $(x,y1)$-$(x,y2)$ 与线段 $(x1,y)$-$(x2,y)$ 相交，那么应该符合 $(x1<x<x2)$、$(y1<y<y2)$，由条件 $(x1<x<x2)$，可以想到先把所有的线段按 X 坐标排序。本题要注意的是，线段在端点重合不算自交，所以 X 轴坐标相同时，事件的顺序要恰当处理。如左图，右端点优先，与 Y 轴平行的线段其次，然后到左端点。

（2）　将 Y 轴表示的区间建立线段树。排序后，每个线段或线段的端点称为一个事件，如左图，S1，S2，S3，S4 分别为一个事件。按 X 坐标由小到大，扫描所有事件，如果遇到平行于 X 轴线段的左端点，则按它的 Y 坐标当成一个点，插入到表示 Y 轴区间的线段树中，如果遇到平行于 X 轴线段的右端点，则把它代表的点从线段树中删除。如果遇到与 Y 轴平行的线段 $L1(x,y1)$-$(x,y2)$，假设有线段 $L2(x1,y)$-$(x2,y)$ 与之相交，由 $(x1<x<x2)$ 可知 $L2$ 的左端点已经被扫描过，那么 $L2$ 代表的点已经插入线段树中；$L2$ 的右端点还没被扫描过，说明 $L2$ 代表的点依然存在于线段树之中。换而言之，只要还在线段树中的点，就满足 $(x1<x<x2)$。要判断 $(y1<y<y2)$ 的条件是否满足，可以在线段树中查找 $[y1+1,y2-1]$（在端点处可以相交）区间内是否有点存在，如果存在，就说明有线段和它相交，该图形不合法。

（3）　具体的实现时要注意线段树每个节点增加一个变量记录该区间内的点数，线段树的单位元(叶子节点)改成一个点，而不是一条单位线段。每次插入和删除后，只要对相关节点的变量进行改动就可以了。插入，删除，及查找过程的伪代码见附录 1。

将 Y 轴坐标离散后，以上过程每次执行的复杂度是 O(logn)级别的，由于所有线段数量是 O(n)级别的，所以整题的复杂度是 O(nlogn)级别。

如果将本题扩展成求这些线段所有交点的个数，则只要在查找时，把区间内的所有点统计出来就可以了。

【例题二】空心长方体[2]

在一个三维正坐标系中，存在 N(N<=5000)个点，现在要求一点 P(x,y,z)，使得 O(0，0，0)与 P(x,y,z)两个顶点构成的长方体内不包括 N 个点中的任何一个点(在长方体边缘不算包括)，并使这个长方体的体积最大。x,y,z 均不得超过 1000000。

【问题分析】

P(x,y,z)代表的长方体包含一点 Q，那么 P 的所有坐标值，都大于 Q 点的坐标值，即 Px>Qx　Py>Qy　Pz>Qz。

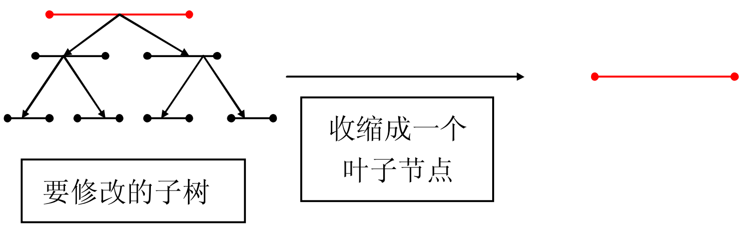
体积最大的长方体，其 P 点任意轴的坐标，都与 N 个点中的一个相同或者和边界相同。假设不同，则把它改成比它大的之中最小的一个，体积增大了，也不会包含任何点。

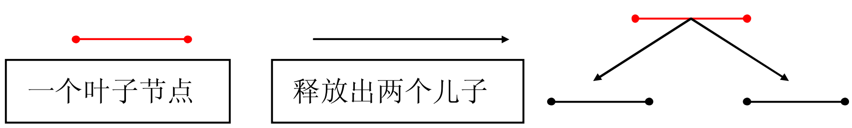
在已经确定 P 的 X 坐标情况下，将所有点的 Y 轴坐标排序，得到序列 y[1]，y[2]，y[3]……。设置数组 max，max[i]记录 P 的 Y 轴坐标为 y[i]时，Z 轴坐标的最大取值，由于 Y 轴坐标增大，Z 轴坐标限制增多，数组 max 的值是单调不增的。

把所有点按 X 轴坐标排序，当 P 点的 X 坐标与第 i 个点相同时，第 i 及第 i 以后的点都不可能被 P 包含了。从小到大枚举 P 点的 X 坐标，这个过程中要不断维护数组 max。因为 P 的 X 坐标由第 i 个点的变成第 i+1 个点的，第 i 个点的坐标就会增加对数组 max 的限制。考虑第 i 个点 Q(x,y,z)增加对数组的限制：

如图，对于 Y 轴坐标比 Qy 小的和 Z 轴坐标比 Qz 小的都不用修改，实际上就是要把上图所表示的区间[i,j]的 max 值都修改成 Qz，而要高效的进行区间操作就可以用到线段树。

建立关于数组 max 的线段树，枚举 P 的 X 轴坐标过程中，每增加一个节点的限制，就相当于修改线段树中的一个区间内的值。修改的区间如果包含节点 V 表示的区间，那么 V 及 V 的儿子都要修改，为避免重复无意义的操作，我们只要修改 V：由于 V 的区间内的值相同，如图，将子树 V 收缩成一个叶子节点。相反的，如图，如果修改的区间不完全包含节点 V，而 V 又已经被收缩成一个叶子节点，那么我们将这个叶子节点释放出两个儿子。





2 Polish Olympiad in Informatics 99, Cuboid，本题考查大块数据的修改

图 2. 子树修改

所求体积为 Px*max[i]*y[i]之一，由于当前 Px，关键 max[i]*y[i]的最大值。线段树中，每个节点要记录其表示的区间内 max[i]*y[i]的最大值，并且在数组被修改时，维护这个最大值。根节点的最大值与 P 点当前 X 坐标相乘,就是当前最大体积。另外，由于 Px，Py，Pz<=1000000，所以要加入(0,0,1000000)、(0,1000000,0)、(1000000,0,0)修改操作的伪代码见附录 2。

【算法总结】

由于利用子树收缩的方法，避免了重复操作。离散后，每次修改的节点数是 O(logn)级别，相关的维护也是这个级别，所以本题的复杂度为 O(nlogn)。

这题的成功之处：不把线段树死板的按看成固定的，而是抓住线段树叶子节点的本质——区间内的各种数据是单一，根据情况把子树收缩为叶子或让叶子释放出儿子，从而避免重复的操作。

【例题三】战场统计系统[3]

2050 年，人类与外星人之间的战争已趋于白热化。就在这时，人类发明出一种超级武器，这种武器能够同时对相邻的多个目标进行攻击。凡是防御力小于或等于这种武器攻击力的外星人遭到它的攻击，就会被消灭。然而，拥有超级武器是远远不够的，人们还需要一个战地统计系统时刻反馈外星人部队的信息。这个艰巨的任务落在你的身上。请你尽快设计出这样一套系统。

这套系统需要具备能够处理如下 2 类信息的能力：

1. 外星人向[x1,x2]内的每个位置增援一支防御力为 v 的部队。
2. 人类使用超级武器对[x1,x2]内的所有位置进行一次攻击力为 v 的打击（防御力不超过 v 的部队都被消灭）。系统需要返回在这次攻击中被消灭的外星人个数。

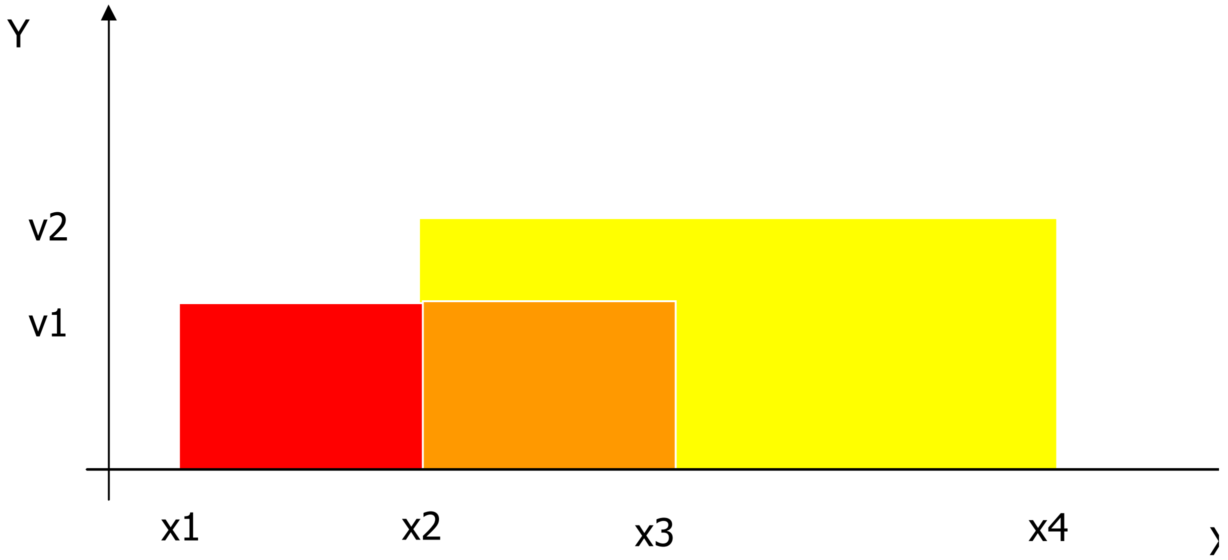
注：防御力为 i 的外星人部队由 i 个外星人组成，其中第 j 个外星人的防御力为 j。

数据范围：1<=x1<=x2<=1000,v<=1000；信息总数 m<=2000。

【问题分析 】

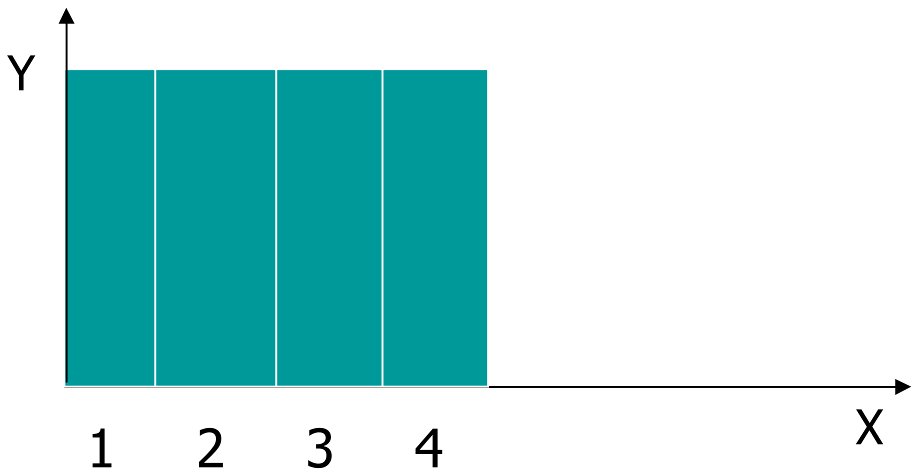
先将两类信息抽象化，我们用二维数组 T[x,y]表示，在单位区间[x,x]上，防御力为 y 的部队数。如果在[x1,x2]增加防御力为 v 的部队，就是在这个二维数组内，添加一个[x1,x2][1,v]的矩形，要使用一次武器就相当于查找一个矩形[x1,x2][1,v]内的部队数目，并将此矩形内的所有部队删除。于是，这题就属于数据处理类型，需要我们比较高效的完成这两个操作：(1)在二维数组内加入一个矩形；(2)查找一个矩形内的部队数并将其删除。如图，插入了一矩形[x1,x3][1,v1]（红色表示），然后删除一矩形[x2,x4][1,v2]（黄色表示，橙色表示删去部分）。

[3] OIBH 练习赛#6, 考查不规则的删除和二维以上的推广



如果采用最简单的方法：对矩形内的每个点进行操作，复杂度高达 $O(m*n*v)$。

首先可以考虑在一维上优化，如图，孤立处理每个单位区间[x,x]，每个单位区间上用一棵线段树记录各种防御力的部队数。



（1）插入一个矩形[$x1$,$x2$][1,v]，就变成了执行 $x2-x1+1$ 次在线段树上插入线段[1,v]的操作，这和一般插入是一样的。

（2）统计一个矩形内的部队数目并删除该矩形[$x1$,$x2$][1,v]内的部队。同样执行 x2-x1+1 次统计并删除线段树区间[1,v]的操作。查找是比较简单的，每个节点记录该区间被覆盖的次数 t，以及该区间的部队总数 total：V.total=V.t*(V.b-V.a+1)+V.Lson.total+V.Rson.total。而删除该区间内的部队就比较复杂，因为删除的区间并不与插入的对应。假设现在要删除节点 V 及它的儿子，则此操作相当与把 V 及它儿子的记录全部修改成 0。这就可以用到上述提到的大规模修改数据的办法——收缩子树。在节点 V 中增加一个布尔变量记录 V 是否是叶子，要删除 V，则把它的布尔变量赋成真，另外如果要在区间 V 内删除区间[1,v]，而[1,v]并未完全覆盖区间 V，那么原来插入的覆盖区间 V 的区间就要被破开，我们就把覆盖 V 的区间分给它的

两个儿子，删除的伪代码见附录 3。相反的，要在区间 V 内插入[1,*v*]，如果区间 V 不被[1,*v*]完全覆盖，且 V 是一个叶子节点，那么就将 V 释放出两个儿子节点，然后再把区间[1,*v*]插入它的两个儿子中去。

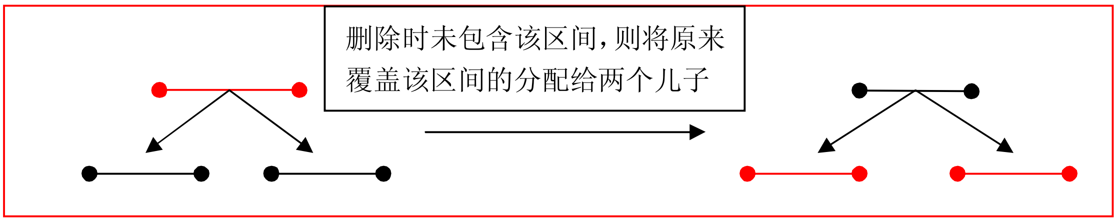


图 3. 特殊的删除

优化后每次删除、统计、插入的复杂度都是 O(n*logv)，所以总的复杂度降为 O(m*n*logv);

【算法改进】

由于这是个二维的区间，不妨将整个二维的区间建成类似线段树的面积树：假设整个二维区间是$[1..2^k][1..2^k]$，T(*x*1,*x*2,*y*1,*y*2)表示二维区间[*x*1,*x*2][*y*1,*y*2]的面积树，递归定义一棵树 T:。

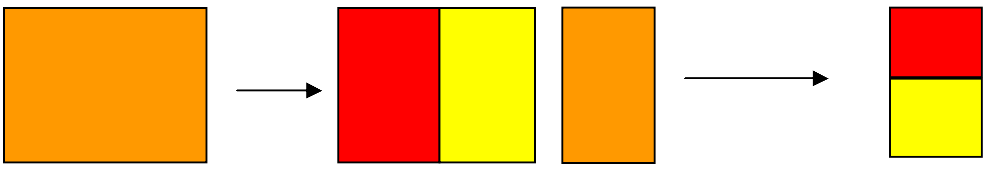
若 T(*x*1,*x*2,*y*1,*y*2)的参数满足 *x*1=*x*2、*y*1=*y*2，T 为叶子节点

若 T(*x*1,*x*2,*y*1,*y*2) 的参数满足 (*x*2-*x*1<=*y*2-*y*1)那么它的左右儿子分别为：

T1(*x*1,(*x*1+*x*2) div 2,*y*1,*y*2)、T2((*x*1+*x*2) div 2+1,*x*2,*y*1,*y*2)

若 T(*x*1,*x*2,*y*1,*y*2) 的参数满足 (*x*2-*x*1>*y*2-*y*1)那么它的左右儿子分别为：

T1(*x*1,*x*2,*y*1,(*y*1+*y*2) div 2)、T2(*x*1,*x*2,(*y*1+*y*2) div 2+1,*y*2)



其实这样建树与大家熟悉四叉树是等价的，只是减少了儿子数目使操作比较方便。由于任意矩形被四叉树树分为不超过 2^k 块，所以这样建树后，任意矩形被面积树分为不超过 2^k 块。所以在这样的一棵树中进行一次插入或删除的操作复杂度不超过 $O(2^k)$,具体操作和一维线段树是类似的，这里不再赘述。所以总的复杂度降到了 O(m*(n+v));

总结

总结这三题可以得出线段树解题的一般方法：

1. 分析题目，知道题目具有很明显的区间性。
2. 根据题目要求将一个区间建成线段树，一般的题目都需要对坐标离散。建树时，不要拘泥于线段树这个名字而只将线段建树，只要是表示区间，而且区间是由单位元素(可以是一个点、线段、或数组中一个值)组成的，都可以建线段树；不要拘泥于一维，根据题目要求可以建立面积树、体积树等等。
3. 树的每个节点根据题目所需，设置变量记录要求的值。
4. 用树形结构来维护这些变量：如果是求总数，则是左右儿子总数之和加

上本节点的总数，如果要求最值，则是左右儿子的最值再联系本区间。利用每次插入、删除时，都只对 O(logL)个节点修改这个特点，在 O(logL)的时间内维护修改后相关节点的变量。

5. 在非规则删除操作，和大规模修改数据操作中，要灵活的运用子树的收缩与叶子节点的释放，避免重复操作。

附录：

1.

```
Procedure  Insert(y, V){把点 y 插入区间 V}
   Begin
     If V.a =V.b then Inc(V.t)
     Else Begin
           If y<=(V.a+V.b) div 2 then Insert(c, d, V.Lson);
           If y> (V.a+V.b) div 2 then Insert(c, d, V.Rson);
           V.t = V.Lson.t+V.Rson.t
         End;
   End;
   Procedure  Delete(y, V){把点 y 从区间 V 删除}
   Begin
     If V.a=V.b then Dec(V.t)
     Else Begin
           If y<=(V.a+V.b) div 2 then Delete(c, d, V.Lson);
           If y> (V.a+V.b) div 2 then Delete(c, d, V.Rson);
           V.t = V.Lson.t+V.Rson.t
         End;
   End;
   Function Find(y1,y2,V) : boolean;{在区间 V 内查找[y1,y2] 区间有没有点}
   Begin
     If y1<=V.a<=V.b<=y2 then find := (V.t<>0);
                     else Begin
                       find := false;
                       if y1<=(V.a+V.b) div 2 then
                         if find(y1,y2,V.Lson) then find := true;
                       if y2>(V.a+V.b) div 2 then
                         if find(y1,y2,V.Rson) then find := true;
                     End;
   End;
```

2.

```
procedure modify(qx, qy, qz, V,a,b);{在区间 V[a,b]内加入限制点(qx, qy, qz))
begin                                      {V.l=ax[V.a],V.r=max[V.b]}
  if (y[a]>qy) and (V.r>qz) then begin(收缩子树 V)
    V.leaf := true;
    V.l := qz; V.r := qz;
```

```
    V.maxS := qz*y[b];
    exit;
  end;
  if (y[b]<=qy) or (V.l<=qz) then
    exit;
  if V.leaf then begin   {将叶子节点V释放出两个儿子，儿子的属性与V一致}
   V.Lson = V;  V.Lson.maxS = y[(a+b) div 2]*V.Lson.l
   V.Rson = V;  V.Rson.maxS = y[b]*V.Rson.l;
   V.leaf = false;
  end;
  modify(qx, qy, qz, V, a, (a+b) div 2);
  modify(qx, qy, qz, V, (a+b) div 2+1,b);
  V.maxS= maximun{V.Lson.maxS,V.Rson.maxS}
end;
```

3.

```
Procedure Delete(y1,y2,V);
Begin
  If y1<=V.a<=V.b<=y2 then
   Begin
     V.leaf := true;             {把V变成叶子节点}
     V.t := 0;
     V.total := 0;
   Else if not V.leaf then Begin {如果是叶子节点说明该区间是空，不用操作}
        Inc(V.Lson.t, V.t);       {把覆盖V的区间分给两个儿子}
        Inc(V.Rson, V.t);
        V.t := 0;
        If y1<=(V.a+V.b) div 2 then Delete(y1,y2,V.Lson);
        If y2>(V.a+V.b) div 2 then Delete(y1,y2,V.Rson);
        V.total := V.Lson.total+V.rson.total;
      End;
End;
```