[**线段树 总结**](http://www.cppblog.com/MemoryGarden/archive/2009/04/11/79565.html)

以下记录，有个人见解，如若有错误，欢迎提出，我一定改正。  
  
附件 ：  [/Files/MemoryGarden/xuemao.rar](http://www.cppblog.com/Files/MemoryGarden/xuemao.rar)    
  
经过了一段时间，终于学会了线段树的基本应用了

线段树的应用非常灵活，我只是学会了皮毛，个人倾向于静态，所以需要以下全局变量。

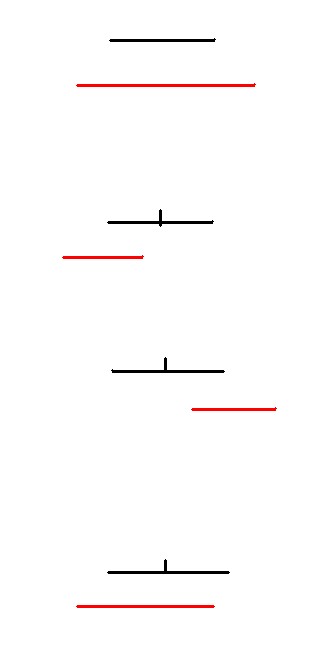
http://www.cppblog.com/Images/OutliningIndicators/None.gifint tree[max], leftchild[max], rightchild[max], leftvalue[max], rightvalue[max], cover[max],node, coverlength;

线段树的最基本的操作：  
  
[1]  建树 ：

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid construct(int left, int right)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int index, mid;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    node++;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    index = node;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    leftvalue[index] = left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    rightvalue[index] = right;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    cover[index] = 0;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    if(left + 1 < right)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        mid = (left + right) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        leftchild[index] = node + 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(left, mid);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        rightchild[index] = node + 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(mid, right);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

上述代码利用二分的方法，建立一棵叶子节点为 (i， i+ 1)的线段树 。 特点是每棵树的根，一定包含于子树的线段。  
  
[2] 插入一段线段：

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid insert(int index, int c, int d)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int mid;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(c <= leftvalue[index] && rightvalue[index] <= d)cover[index]++;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(leftvalue[index] + 1 < rightvalue[index])http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            mid = (leftvalue[index] + rightvalue[index]) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(c < mid)insert(leftchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(d > mid)insert(rightchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

解释 ：当要插入的线段[c, d] 覆盖了线段树上的某段线段的时候，将其标记覆盖的cover[]数组自加。如果不覆盖该线段，分为三种情况，可自己画           一下，再根据当前线段于要插入线段的关系，分情况递归左右子树  
  
  
上图1 为覆盖的情况, 运行

http://www.cppblog.com/Images/OutliningIndicators/None.gifif(c <= leftvalue[index] && rightvalue[index] <= d)cover[index]++;

上图2 为 c < mid  && d < mid   运行

http://www.cppblog.com/Images/OutliningIndicators/None.gifif(c < mid)insert(leftchild[index], c, d);

上图3 为 c > mid && d > mid  运行

http://www.cppblog.com/Images/OutliningIndicators/None.gifif(d > mid)insert(rightchild[index], c, d);

上图4 为 c < mid && d > mid 运行

http://www.cppblog.com/Images/OutliningIndicators/None.gifif(c < mid)insert(leftchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/None.gifif(d > mid)insert(rightchild[index], c, d);

[3]删除一段线段：

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid Delete(int index, int c, int d)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int mid;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(c <= leftvalue[index] && rightvalue[index] <= d && cover[index])  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        cover[index]--;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(leftvalue[index] + 1 < rightvalue[index])http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            mid = (leftvalue[index] + rightvalue[index]) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(c < mid)Delete(leftchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(d > mid)Delete(rightchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif

与插入一样理解。  
  
[4] 统计：  
  
关于统计，有很多，也很灵活，对于每个题目来说，线段树内每个节点内的要表示的有很多，很灵活，这里只说说测度，也就是最后被覆盖的线段的总长度

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid count(int index)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(cover[index] > 0)total += y[rightvalue[index]] - y[leftvalue[index]];  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(leftvalue[index] + 1 < rightvalue[index])http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            count(leftchild[index]);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            count(rightchild[index]);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

理解 ：既然某段线段的cover[] 是正数，那么，它的子树也一定被cover，因为前面说过的“特点是每棵树的根，一定包含于子树的线段”  
所以长度 = 右边界 - 左边界；  
  
以上为线段树的最基本的操作，是对长度的操作。  
  
对于线段树的应用，还有一种比较广泛，就是类似于涂色的问题 ：  
  
很多大大们都介绍过了， 薛茅的论文上讲解的也非常明白   
  
这类问题，大多会有一个bj[]；数组；  
  
这个数组的用处，是当要插入的线段覆盖了线段树的某一节点段时候，拿染色举例，那么以当前节点位根的子树，都会有相同的颜色，所以我们对它的左右儿子做上标记，在每次对线段树上的节点操作的时候，先检查节点的bj[]数组是否被标记过，如若标记过，那么以当前节点为根的子树的颜色都应该是一样的，也就是bj的颜色，然后把当前节点的bj清楚，当前节点的两个儿子设上标记。 说的笨了些，可以参考我上传得pdf && ppt;  
  
对于染色类型的线段树的操作 ：  
需要以下的全局变量 只增加了一个bj[]

http://www.cppblog.com/Images/OutliningIndicators/None.gifint tree[max], leftchild[max], rightchild[max], leftvalue[max], rightvalue[max], color[max], bj[max];  
http://www.cppblog.com/Images/OutliningIndicators/None.gifint node = 0;

[1] 树的建立 ：

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid construct(int left, int right)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int mid, index;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    node++;index = node;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    leftvalue[index] = left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    rightvalue[index] = right;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    color[index] = -2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bj[index] = -1; //代表未设过标记  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    if(left + 1 < right)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        mid = (left + right) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        leftchild[index] = node + 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(left, mid);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        rightchild[index] = node + 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(mid, right);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

[2] 处理 bj 节点的clear函数

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid clear(int index)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    color[index] = bj[index];  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bj[leftchild[index]] = bj[index];  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bj[rightchild[index]] = bj[index];  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bj[index] = -1;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

仅以颜色举例  
  
[3]树的插入 ：

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid insert(int index, int c, int d, int x)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int mid;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(bj[index] != -1)clear(index);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(color[index] == x)return;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(c <= leftvalue[index] && rightvalue[index] <= d)  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        bj[index] = x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(leftvalue[index] + 1 < rightvalue[index])http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            color[index] = -1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            mid = (leftvalue[index] + rightvalue[index]) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(c < mid)insert(leftchild[index], c, d, x);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(d > mid)insert(rightchild[index], c, d, x);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

解释 ：如果此节点被标记，则当前节点的bj删除，然后两个子节点设上bj  
  
[4] 删除类似  
  
举例 ：[Count the Colors](http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1610)  
  
按照上述所说，操作即可 代码如下 ：

http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <stdio.h>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <string.h>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#define max 16000  
http://www.cppblog.com/Images/OutliningIndicators/None.gifint tree[max], leftchild[max], rightchild[max], leftvalue[max], rightvalue[max], color[max], bj[max];  
http://www.cppblog.com/Images/OutliningIndicators/None.gifint node = 0, temp, res[max];  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid clear(int index)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    color[index] = bj[index];  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bj[leftchild[index]] = bj[index];  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bj[rightchild[index]] = bj[index];  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bj[index] = -1;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid construct(int left, int right)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int mid, index;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    node++;index = node;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    leftvalue[index] = left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    rightvalue[index] = right;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    color[index] = -2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bj[index] = -1; //代表未设过标记  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    if(left + 1 < right)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        mid = (left + right) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        leftchild[index] = node + 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(left, mid);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        rightchild[index] = node + 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(mid, right);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid insert(int index, int c, int d, int x)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int mid;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(bj[index] != -1)clear(index);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(color[index] == x)return;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(c <= leftvalue[index] && rightvalue[index] <= d)  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        bj[index] = x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(leftvalue[index] + 1 < rightvalue[index])http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            color[index] = -1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            mid = (leftvalue[index] + rightvalue[index]) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(c < mid)insert(leftchild[index], c, d, x);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(d > mid)insert(rightchild[index], c, d, x);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid search(int index)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //printf("search %d \n", index);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(bj[index] != -1)clear(index);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    if(color[index] != -1)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(color[index] != temp)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(color[index] >= 0)  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                res[color[index]]++;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            temp = color[index];  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    }elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        search(leftchild[index]);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        search(rightchild[index]);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifint main()http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    freopen("(a,a+1)segement-color.in", "r", stdin);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    freopen("(a,a+1)segement-color.out", "w", stdout);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int i, n, c, d, x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    while(scanf("%d", &n) != -1)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        node = 0;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        memset(res, 0, sizeof(res));  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        memset(leftchild, 0, sizeof(leftchild));  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        memset(rightchild, 0, sizeof(leftchild));  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(0, 6);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        for(i = 0; i < n; i++)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            scanf("%d%d%d", &c, &d, &x);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            insert(1, c, d, x);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        search(1);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        for(i = 0; i < 8001; i++)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(res[i])  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                printf("%d %d\n", i, res[i]);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        puts("");  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    return 0;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif

关于计算面积的线段树的应用 + 离散化  
  
对于离散化，我的理解也不是很清楚 ，不知道理解的对不对。  
  
如果题目的范围很大，而有效的数据又很少，我们可以将有效数据排序，重新建立坐标轴，然后以有效数据的个数作为范围；  
  
比如有整数类型点 4 5 6 7 则 离散后的数组为 f[0] = 4   f[1] = 5  f[2] = 6 f[3] = 7;  
或者浮点类型 4.5    5.6    6.54   7.59 离散后的数组为 f[0] = 4.5  f[1] = 5.6 f[2] = 6.54  f[3] = 7.59;  
  
这样我们就把浮点类型，或者整数类型的有效数据转映射到正数下标的数组里面去了。  
  
我们可以建立一棵(0, 4)的线段树 当插入一条线段 [0， 1] 代表我们插入了[4---5]  或者 [4.5---5.6] 这样一条线段，由于我们将有效数据排序，所以f数组的下表与有效数据的同为递增的关系，所以当我们插入[0， 3]这条线段的时候，也代表了我们插入了[4---7] 或者[4.5---7.59]这样一条线段，[0， 3] 覆盖了[0， 1]这条线段，同样，[4---7] 也会覆盖 [4---5]  或者[4.5---7.59] 也会覆盖[4.5---5.6] 。  
  
以上就是我对离散化的理解了。  
  
举例一题 ：[Atlantis](http://acm.jlu.edu.cn/joj/showproblem.php?pid=1043)   
  
这个题目我们就把横纵坐标都进行离散，然后对于纵坐标，用线段树操作。  
  
把竖直的线段离散后，存储在一个结构体中，存储x坐标, 和上下两端的y坐标，kind用于标记对于一个矩形来说，左边的竖线，还是右边竖线

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifstruct Lhttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    double x, y1, y2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bool kind;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}line[max];  
http://www.cppblog.com/Images/OutliningIndicators/None.gif

把纵坐标离散，然后排序。按照我上面离散化的意思，建树。  
  
我用的是<map> 能标记重复，也能排序

http://www.cppblog.com/Images/OutliningIndicators/None.gifmap<double, int> Y;  
http://www.cppblog.com/Images/OutliningIndicators/None.gifmap<double, int>::iterator it;  
http://www.cppblog.com/Images/OutliningIndicators/None.gif

对line 数组按照x坐标排序，对于line数组，逐2个计算横坐标的差，总坐标若为左竖线，则插入，若为右竖线，则删除。。。。说得太不明白了，连我自己都不明白。还是看看代码吧  请大家自己理解。  
  
代码写的不好，在poj上不是0ms。

http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <stdio.h>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <string.h>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <algorithm>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <map>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <math.h>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#define max 401  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#define eps 1e-8  
http://www.cppblog.com/Images/OutliningIndicators/None.gifusing namespace std;  
http://www.cppblog.com/Images/OutliningIndicators/None.gifint tree[max], leftchild[max], rightchild[max], rightvalue[max], leftvalue[max], cover[max];  
http://www.cppblog.com/Images/OutliningIndicators/None.gifint node, linelength;  
http://www.cppblog.com/Images/OutliningIndicators/None.gifdouble y[max \* 2], total;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifstruct Lhttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    double x, y1, y2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    bool kind;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}line[max];  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifmap<double, int> Y;  
http://www.cppblog.com/Images/OutliningIndicators/None.gifmap<double, int>::iterator it;  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifbool cmp(L a, L b)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    return a.x < b.x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid construct(int left, int right)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int index, mid;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    node++;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    index = node;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    leftvalue[index] = left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    rightvalue[index] = right;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    cover[index] = 0;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    if(left + 1 < right)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        mid = (left + right) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        leftchild[index] = node + 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(left, mid);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        rightchild[index] = node + 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(mid, right);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid insert(int index, int c, int d)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int mid;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(c <= leftvalue[index] && rightvalue[index] <= d)cover[index]++;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(leftvalue[index] + 1 < rightvalue[index])http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            mid = (leftvalue[index] + rightvalue[index]) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(c < mid)insert(leftchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(d > mid)insert(rightchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid Delete(int index, int c, int d)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int mid;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(c <= leftvalue[index] && rightvalue[index] <= d && cover[index])  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        cover[index]--;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(leftvalue[index] + 1 < rightvalue[index])http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            mid = (leftvalue[index] + rightvalue[index]) / 2;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(c < mid)Delete(leftchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(d > mid)Delete(rightchild[index], c, d);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid count(int index)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if(cover[index] > 0)total += y[rightvalue[index]] - y[leftvalue[index]];  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    elsehttp://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if(leftvalue[index] + 1 < rightvalue[index])http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            count(leftchild[index]);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            count(rightchild[index]);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifint main()http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    freopen("1043.in", "r", stdin);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    freopen("1043.out", "w", stdout);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int n, i, j, num, c, d, o = 1;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    double lx, ly, rx, ry, area;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    while(scanf("%d", &n), n)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        linelength = node = num = 0;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        Y.clear();area = total = 0.0;   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        for(i = 0; i < n; i++)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            scanf("%lf%lf%lf%lf", &lx, &ly, &rx, &ry);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            line[linelength].x = lx;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            line[linelength].y1 = ly;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            line[linelength].kind = true;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            line[linelength++].y2 = ry;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            line[linelength].x = rx;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            line[linelength].y1 = ly;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            line[linelength].kind = false;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            line[linelength++].y2 = ry;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            it = Y.find(ly);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(it == Y.end())Y.insert(make\_pair(ly, num++));  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            it = Y.find(ry);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(it == Y.end())Y.insert(make\_pair(ry, num++));  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        for(it = Y.begin(), i = 0; it != Y.end(); it++, i++)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            y[i] = it->first;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            it->second = i;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        sort(line, line + linelength, cmp);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        construct(0, num);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        for(i = 0; i < linelength - 1; i++)http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            c = Y.find(line[i].y1)->second;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            d = Y.find(line[i].y2)->second;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if(line[i].kind)insert(1, c, d);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            else Delete(1, c, d);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            total = 0.0;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            count(1);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            area += total \* (line[i + 1].x - line[i].x);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        printf("Test case #%d\nTotal explored area: %.2lf\n\n", o++, area);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    return 0;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif

ps ：切记 如果要建立一棵[0， a] 的一棵线段树，我们需要2 \* a 个节点。 我因为这个wa了n次。  
  
  
  
这是我所有学过的线段树的知识，还会继续学习，如果有毛病，请大家指出，菜鸟在这里谢过；