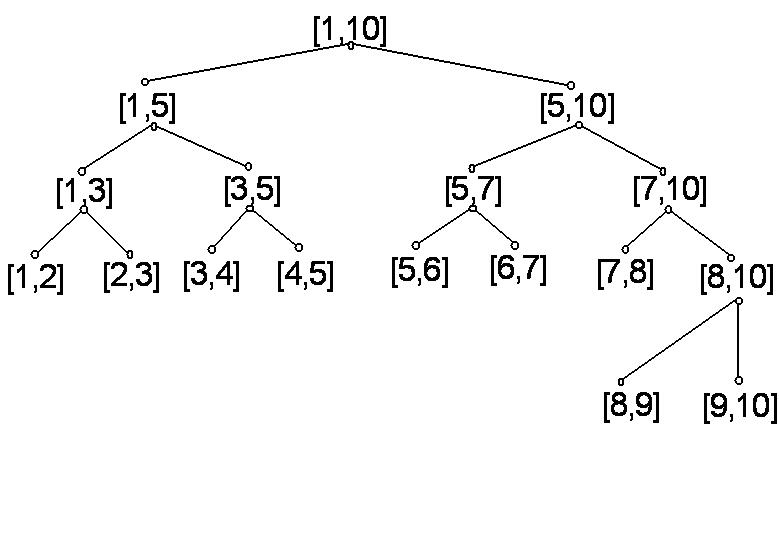
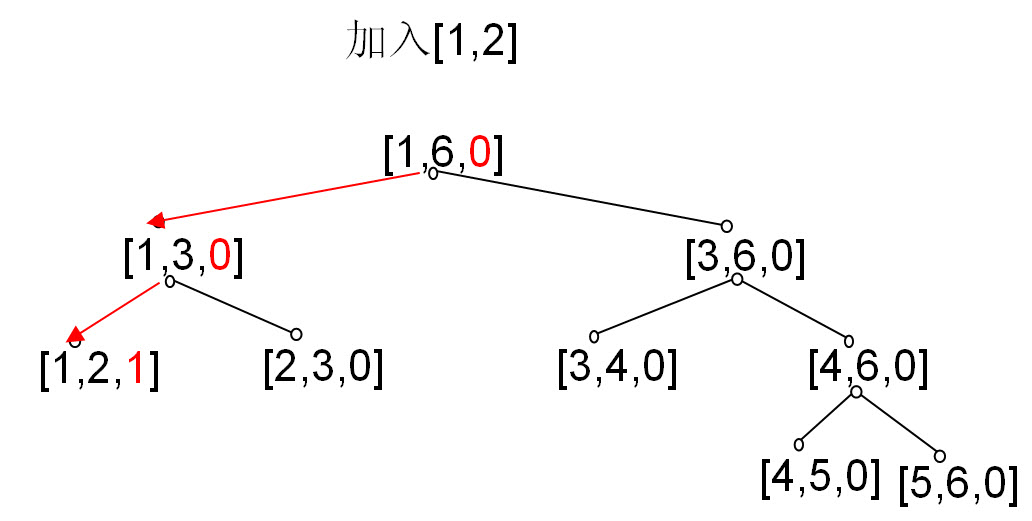
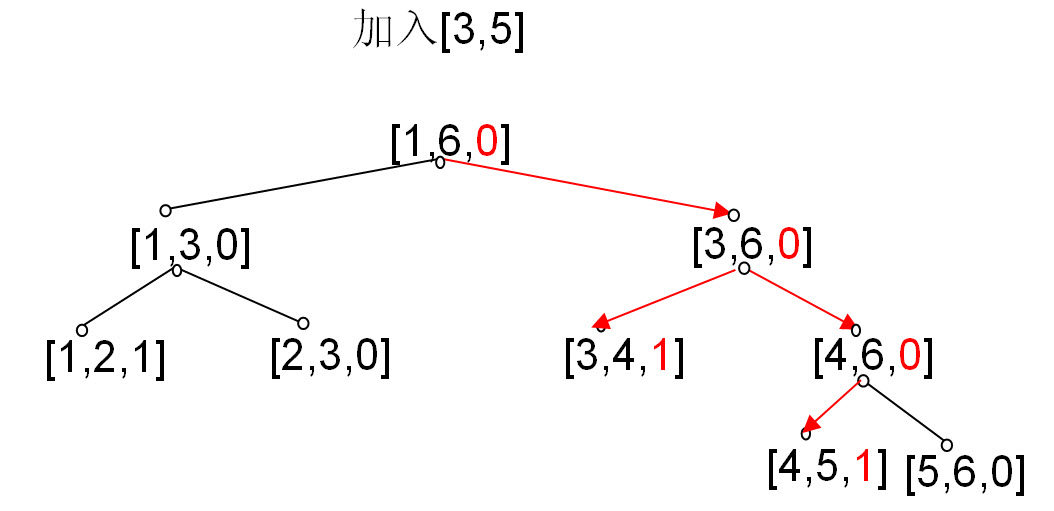
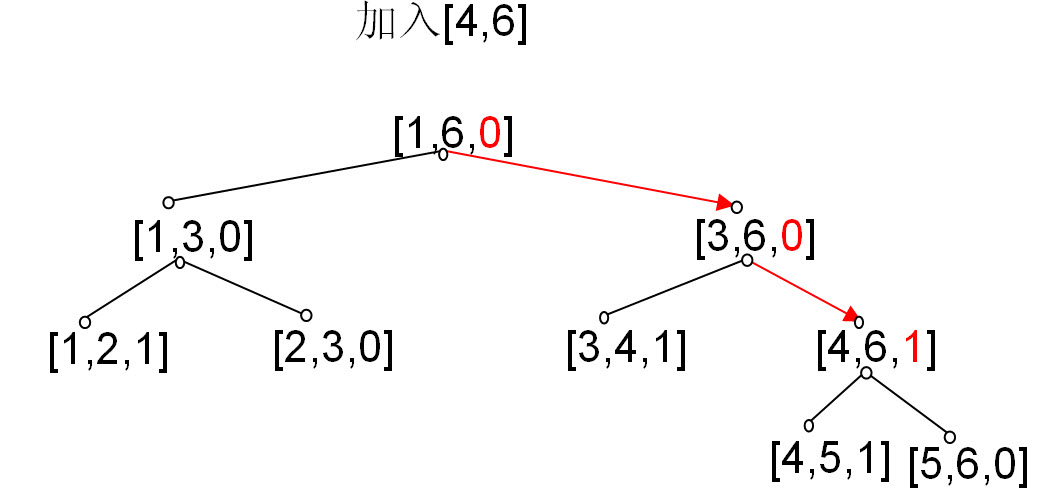
**[线段树](http://kmplayer.javaeye.com/blog/576486)**

**文章分类:**[**C++编程**](http://www.javaeye.com/blogs/category/cpp)

**线段树的构造思想**   
线段树是一棵二叉树，树中的每一个结点表示了一个**区间**[a,b]。每一个叶子节点表示了一个单位区间。对于每一个非叶结点所表示的结点[a,b]，其左儿子表示的区间为[a,(a+b)/2]，右儿子表示的区间为[(a+b)/2,b]。   
例如:   
  
**线段树的运用**   
线段树的每个节点上往往都增加了一些其他的域。在这些域中保存了某种动态维护的信息，视不同情况而定。这些域使得线段树具有极大的灵活性，可以适应不同的需求。   
  
例1:求覆盖线段的总长度   
[10000,22000]   [30300,55000]   [44000,60000]   [55000,60000]   
排序得10000，22000，30300，44000，55000，60000   
对应得 1，  2，  3，  4，  5，  6   
   [1,2]   [3,5]   [4,6]     [5,6]   
  
线段树做法:   
  
  
  
  
  


例2:http://acm.pku.edu.cn/JudgeOnline/problem?id=2528

题意:贴海报,输出可以看到的个数.

Java代码

//贴海报,输出没有被覆盖的个数

//可以改成cpp试试

#include<stdio.h>

struct node

{

int l,r; //左右孩子的编号

int st,mi,en;

int id;

}; // 线段树简单一维

const int maxN = 50000002; //线段树的节点个数

const int maxL = 10000020; //台阶的宽度上限

node segment\_tree[maxN]; //保存着线段树的所有节点

#define tree segment\_tree

int root, ptr;

void insert(int cr, int start, int end, int color) //插入到指定区域,同时初始化沿途的所有节点

{

if(start >= end) //不符合输入要求

return;

if(tree[cr].st == start && tree[cr].en == end) //输入区间正好等于节点的表示区间

{

tree[cr].id = color; //这个区间属于该海报

return;

}

int mid = (tree[cr].st + tree[cr].en) / 2;

if(tree[cr].l == 0) //意味着还没有初始化孩子结点

{

//ptr代表节点的编号

tree[cr].l = ptr++;

tree[tree[cr].l].l = tree[tree[cr].l].r = 0;

tree[tree[cr].l].id = -1;

tree[tree[cr].l].st = tree[cr].st, //这里对左右孩子的范围进行初始化

tree[tree[cr].l].en = mid;

}

if(tree[cr].r == 0)

{

tree[cr].r = ptr++;

tree[tree[cr].r].l = tree[tree[cr].r].r = 0;

tree[tree[cr].r].id = -1;

tree[tree[cr].r].st = mid,

tree[tree[cr].r].en = tree[cr].en;

}

if(tree[cr].id != 0) //之后的子区间肯定都属于该海报

{

tree[tree[cr].l].id = tree[tree[cr].r].id = tree[cr].id;

tree[cr].id = 0;

}

if(start >= mid){

insert(tree[cr].r, start, end, color);

return;

}

if(end <= mid){

insert(tree[cr].l, start, end, color);

return;

}

insert(tree[cr].l, start, mid, color);

insert(tree[cr].r, mid, end, color);

}

char exist[10001];

void trail(int cr) //统计可以看见的节点编号

{

if(cr == 0 || tree[cr].id == -1)

return;

exist[tree[cr].id] = 1; //id不为0,意味着只有它可见,但之后的节点都看不见了.

if(tree[cr].id != 0) //不为0意味着后面的区域都要被覆盖.

return;

trail(tree[cr].l);

trail(tree[cr].r);

}

//初始化跟节点

void init()

{

root = 1;

tree[root].l = tree[root].r = tree[root].id = 0;

tree[root].st = 1, tree[root].en = maxL, tree[root].mi = (1 + maxL)/2;

ptr = 2;

}

int main()

{

int test,n,i,l,r;

scanf("%d", &test);

while(test--)

{

init();

scanf("%d",&n);

for(i = 1; i <= n; i++)

{

scanf("%d%d",&l,&r);

insert(1, l, r+1, i); //从根节点开始插入

}

for(i = 1; i <= n; i++)

exist[i] = 0;

trail(1);

int ans = 0;

for(i = 1; i <= n; i++)

if(exist[i])

ans++;

printf("%d\n",ans);

}

return 0;

}