线段树。  
父亲节点的改变，需要让儿子节点继承父亲节点原来的状态，而儿子节点的改变也可以更新父亲节点的信息。  
Hotel  
<http://acm.pku.edu.cn/JudgeOnline/problem?id=1823>

http://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifhttp://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifPOJ 1823  
//flag1=1表示没人，flag0=0表示人满  
//每个节点记录左端点，右端点，左端点开始连续free的最大值lmax，右端点开始连续free的最大值rmax，节点的连续free的最大值ma，以及两个flag  
#include<iostream>  
#include<cmath>  
#include<algorithm>  
using namespace std;  
struct node  
{  
    int l,r;  
    int lmax,rmax,ma;  
    bool flag1,flag0;//flag1表示没人，flag0表示有人  
}T[160001];  
int build(int s,int e,int u)  
{  
    T[u].l=s;  
    T[u].r=e;  
    T[u].flag0=0;//非满  
    T[u].flag1=1;//全空  
    T[u].lmax=T[u].rmax=T[u].ma=e-s+1;  
    if(s<e)  
    {  
        int mid=(s+e)>>1;  
        build(s,mid,2\*u);  
        build(mid+1,e,2\*u+1);  
    }  
    return 0;  
}  
int checkin(int s,int e,int u)//住人  
{  
    if(T[u].flag1)  
    {  
        T[u].flag1=0;//非空  
        T[2\*u].flag1=1;  
        T[2\*u].flag0=0;  
        T[2\*u].lmax=T[2\*u].rmax=T[2\*u].ma=(T[2\*u].r-T[2\*u].l+1);  
        T[2\*u+1].flag1=1;  
        T[2\*u+1].flag0=0;  
        T[2\*u+1].lmax=T[2\*u+1].rmax=T[2\*u+1].ma=(T[2\*u+1].r-T[2\*u+1].l+1);  
    }  
    if(s<=T[u].l&&e>=T[u].r)  
    {  
        T[u].flag0=1;//住满  
        T[u].lmax=T[u].rmax=T[u].ma=0;  
        return 0;  
    }  
    if(s<=T[2\*u].r)  
    {  
        checkin(s,e,2\*u);  
    }  
    if(e>=T[2\*u+1].l)  
    {  
        checkin(s,e,2\*u+1);  
    }  
    T[u].ma=max(T[2\*u].ma,T[2\*u+1].ma);  
    T[u].ma=max(T[u].ma,T[2\*u].rmax+T[2\*u+1].lmax);  
    if(T[2\*u].lmax==T[2\*u].r-T[2\*u].l+1)  
    {  
        T[u].lmax=T[2\*u].lmax+T[2\*u+1].lmax;  
    }  
    else  
        T[u].lmax=T[2\*u].lmax;  
    if(T[2\*u+1].rmax==T[2\*u+1].r-T[2\*u+1].l+1)  
    {  
        T[u].rmax=T[2\*u].rmax+T[2\*u+1].ma;  
    }  
    else  
        T[u].rmax=T[2\*u+1].rmax;  
    if(T[u].ma==0)  
        T[u].flag0=1;//住满  
    return 0;  
}  
int checkout(int s,int e,int u)//走人  
{  
    if(T[u].flag0)//住满  
    {  
        T[u].flag0=0;//非满  
        T[2\*u].flag0=1;  
        T[2\*u].flag1=0;  
        T[2\*u].lmax=T[2\*u].rmax=T[2\*u].ma=0;  
        T[2\*u+1].flag0=1;  
        T[2\*u+1].flag1=0;  
        T[2\*u+1].lmax=T[2\*u+1].rmax=T[2\*u+1].ma=0;  
    }  
    if(s<=T[u].l&&e>=T[u].r)  
    {  
        T[u].flag1=1;//全走，为空  
        T[u].lmax=T[u].rmax=T[u].ma=T[u].r-T[u].l+1;  
        return 0;  
    }  
    if(s<=T[2\*u].r)  
        checkout(s,e,2\*u);  
    if(e>=T[2\*u+1].l)  
        checkout(s,e,2\*u+1);  
    T[u].ma=max(T[2\*u].ma,T[2\*u+1].ma);  
    T[u].ma=max(T[u].ma,T[2\*u].rmax+T[2\*u+1].lmax);  
    if(T[2\*u].lmax==T[2\*u].r-T[2\*u].l+1)  
    {  
        T[u].lmax=T[2\*u].lmax+T[2\*u+1].lmax;  
    }  
    else  
        T[u].lmax=T[2\*u].lmax;  
    if(T[2\*u+1].rmax==T[2\*u+1].r-T[2\*u+1].l+1)  
    {  
        T[u].rmax=T[2\*u].rmax+T[2\*u+1].rmax;  
    }  
    else  
        T[u].rmax=T[2\*u+1].rmax;  
    if(T[u].ma==T[u].r-T[u].l+1)  
        T[u].flag1=1;  
    return 0;  
}  
int n,k;  
int a,b,c;  
int main()  
{  
    scanf("%d%d",&n,&k);  
    build(1,n,1);  
    int i,j;  
    for(i=1;i<=k;i++)  
    {  
        scanf("%d",&c);  
        if(c==1)  
        {  
            scanf("%d%d",&a,&b);  
            checkin(a,a+b-1,1);  
        }  
        else if(c==2)  
        {  
            scanf("%d%d",&a,&b);  
            checkout(a,a+b-1,1);  
        }  
        else  
            printf("%d\n",T[1].ma);  
    }  
    system("pause");  
    return 0;  
}

<http://acm.pku.edu.cn/JudgeOnline/problem?id=2828>  
Buy Tickets

http://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifhttp://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifPOJ 2828  
//按输入的倒序进行操作，可知按倒序排列，插入的位置是最终的位置。  
如果插入的位置前有pos-1个空位，则可以插入。后则插在后面  
#include<iostream>  
using namespace std;  
struct node  
{  
    int l,r,c;  
}nod[600010];  
int a[200001],b[200001],ans[200001];  
int build(int s,int e,int u)  
{  
    nod[u].l=s;  
    nod[u].r=e;  
    nod[u].c=e-s+1;  
    if(s<e)  
    {  
        int mid=(s+e)>>1;  
        build(s,mid,2\*u);  
        build(mid+1,e,2\*u+1);  
    }  
    return 0;  
}  
int solve(int pos,int val,int u)  
{  
    nod[u].c--;  
    if(nod[u].l==nod[u].r)  
    {  
        ans[nod[u].l]=val;  
        return 0;  
    }  
    if(nod[2\*u].c>=pos)  
        solve(pos,val,2\*u);  
    else  
        solve(pos-nod[2\*u].c,val,2\*u+1);  
    return 0;  
  
  
}  
int main()  
{  
    int n;  
    while(scanf("%d",&n)!=EOF)  
    {  
        int i;  
        build(1,n,1);  
        for(i=1;i<=n;i++)  
        {  
            scanf("%d%d",&a[i],&b[i]);  
            a[i]++;  
        }  
        for(i=n;i>=1;i--)  
            solve(a[i],b[i],1);  
        for(i=1;i<n;i++)  
            printf("%d ",ans[i]);  
        printf("%d\n",ans[i]);  
    }  
    return 0;  
}

<http://acm.pku.edu.cn/JudgeOnline/problem?id=3277>City Horizon

http://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifhttp://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifPOJ 3277  
//离散化+线段树  
//按高度排序后可以直接覆盖掉之前高度小的  
#include<iostream>  
#include<cmath>  
#include<algorithm>  
using namespace std;  
struct Node  
{  
    long long l,r,h,mh;  
}t[310001];  
int map[40100][3];  
int tmp[80202];  
int n;  
long long sum=0;  
int find(int t)  
{  
    int l=1,r=tmp[0];  
    int mid;  
    while(l<=r)  
    {  
        mid=(l+r)>>1;  
        if(tmp[mid]==t)  
            return mid;  
        if(tmp[mid]>t)  
            r=mid-1;  
        else  
            l=mid+1;  
    }  
    return mid;  
}  
int build(int s,int e,int u)  
{  
    t[u].l=s;  
    t[u].r=e;  
    t[u].h=0;  
    t[u].mh=0;  
    if(s+1<e)  
    {  
        int mid=(s+e)>>1;  
        build(s,mid,2\*u);  
        build(mid,e,2\*u+1);  
    }  
    return 0;  
}  
int insert(int s,int e,long long h,int u)  
{  
    if(h<t[u].h)  
        return 0;  
    if(s<=t[u].l&&e>=t[u].r)  
    {  
        if(t[u].h>=0)  
        {  
            t[u].h=h;  
            t[u].mh=h;  
            return 0;  
        }  
        else  
        {      
            if(h>=t[u].mh)  
            {  
                t[u].h=h;  
                t[u].mh=h;  
                return 0;  
            }  
        }  
    }  
    if(t[u].h>=0)  
    {  
        t[2\*u].h=t[u].h;  
        t[2\*u].mh=t[u].mh;  
        t[2\*u+1].h=t[u].h;  
        t[2\*u+1].mh=t[u].mh;  
        t[u].h=-1;  
    }  
    if(s<t[2\*u].r)  
        insert(s,e,h,2\*u);  
    if(e>t[2\*u+1].l)  
        insert(s,e,h,2\*u+1);  
    t[u].mh=max(t[2\*u].mh,t[2\*u+1].mh);  
    return 0;  
}  
int check(int s,int e,int u)  
{  
    if(s<=t[u].l&&e>=t[u].r)  
    {  
        if(t[u].h>=0)  
        {  
            sum+=(tmp[t[u].r]-tmp[t[u].l])\*t[u].h;  
            return 0;  
        }  
    }  
    if(s<t[2\*u].r)  
        check(s,e,2\*u);  
    if(e>t[2\*u+1].l)  
        check(s,e,2\*u+1);  
    return 0;  
}  
  
int cmp(const void\* lhs, const void\* rhs)     
{     
    return ((int\*)lhs)[2] - ((int\*)rhs)[2];     
}     
int main()  
{  
    scanf("%d",&n);  
    int i,j;  
    tmp[0]=0;  
    for(i=1;i<=n;i++)  
    {  
        for(j=0;j<3;j++)  
        {  
            scanf("%d",&map[i][j]);  
            if(j<2)  
                tmp[++tmp[0]]=map[i][j];  
        }  
    }  
    sort(tmp+1,tmp+2\*n+1);  
    tmp[0]=0;  
    for(i=1;i<=2\*n;i++)  
    {  
        if(tmp[i]!=tmp[i-1])  
            tmp[++tmp[0]]=tmp[i];  
    }  
    build(1,tmp[0],1);  
    qsort(map+1, n, 3\*sizeof(int), cmp);   
    for(i=1;i<=n;i++)  
        insert(find(map[i][0]),find(map[i][1]),map[i][2],1);  
    check(1,tmp[0],1);  
    printf("%lld\n",sum);  
    /\*printf("%d\n",tmp[0]);  
    for(i=1;i<=tmp[0];i++)  
    {  
        printf("%d %d\n",tmp[i],find(tmp[i]));  
    }\*/  
    system("pause");  
    return 0;  
}

<http://acm.pku.edu.cn/JudgeOnline/problem?id=3368>  
Frequent Values

http://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifhttp://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifPOJ 3268  
//对于[a,b][b+1,c],若f[b]==b[b+1],则ma=max([a,b],[b+1,c],[a,b][b+1,c]);  
#include<iostream>  
#include<cmath>  
using namespace std;  
struct node  
{  
    int l,r,lmax,rmax,ma;  
}t[400000];  
int a[100001];  
int sum;  
int build(int s,int e,int u)  
{  
    t[u].l=s;  
    t[u].r=e;  
    if(s==e)  
    {  
        t[u].lmax=t[u].rmax=t[u].ma=1;  
        return 0;  
    }  
    int mid=(s+e)>>1;  
    build(s,mid,2\*u);  
    build(mid+1,e,2\*u+1);  
    t[u].ma=max(t[2\*u].ma,t[2\*u+1].ma);  
    t[u].lmax=t[2\*u].lmax;  
    t[u].rmax=t[2\*u+1].rmax;  
    if(a[t[2\*u].r]==a[t[2\*u+1].l])  
    {      
        int sum=t[2\*u].rmax+t[2\*u+1].lmax;  
        t[u].ma=max(t[u].ma,sum);  
        if(a[t[u].l]==a[t[2\*u].r])  
            t[u].lmax=sum;  
        if(a[t[u].r]==a[t[2\*u+1].l])  
            t[u].rmax=sum;  
    }  
    return 0;  
}  
int check(int s,int e,int u)  
{  
    if(s==t[u].l&&e==t[u].r)  
    {  
        return t[u].ma;  
    }  
    if(e<=t[2\*u].r)  
        return    check(s,e,2\*u);  
    if(s>=t[2\*u+1].l)  
    {  
        return check(s,e,2\*u+1);  
    }  
    int q1,q2;  
    q1=check(s,t[2\*u].r,2\*u);  
    q2=check(t[2\*u+1].l,e,2\*u+1);  
    int tmp=max(q1,q2);  
    if(a[t[2\*u].r]==a[t[2\*u+1].l])  
    {  
        tmp=max(tmp,min(t[2\*u].rmax,t[2\*u].r-s+1)+min(t[2\*u+1].lmax,e-t[2\*u+1].l+1));  
    }  
    return tmp;  
}  
int n,q;  
int main()  
{  
      
    while(scanf("%d",&n),n)  
    {  
        scanf("%d",&q);  
        int i;  
        int x,y;  
        for(i=1;i<=n;i++)  
        {  
            scanf("%d",&a[i]);  
        }  
        build(1,n,1);  
        for(i=1;i<=q;i++)  
        {  
            sum=0;  
            scanf("%d%d",&x,&y);  
            printf("%d\n",check(x,y,1));  
        }  
    }  
    system("pause");  
    return 0;  
      
}