POJ 1765 November Rain

二维平面上给出若干屋檐（线段），问各个屋檐流过的水量。看样例就会明白啦，单位横坐标长度降雨量是 1，雨会从高的一端流向低的一端被其他屋檐或地面接住。

所以离散化扫描线暴力搞一搞就好（因为一点上空最多 25 个屋檐）。扫描过程处理屋檐间的关系（给接水屋檐向流水屋檐连边）和计入每一段最高屋檐接到的水。最后在DAG上dfs一下算好答案就可以了。

```
#include<cstdio>
#include<vector>
#include<algorithm>
#include<cstring>
#define pii pair<int,int>
#define ll long long
#define pb push_back
using namespace std;
const int N=40400;
int num[N*2],tot;
struct Point
{
    int x,y;
    Point(int x=0,int y=0):x(x),y(y){}
    Point operator -(Point a){return Point(x-a.x,y-a.y);}
};
struct segment
{
    Point a,b;
    double k;
    int id;
    void init(){scanf("%d%d%d%d",&a.x,&a.y,&b.x,&b.y),k=(b.y-a.y)*1.0/(b.x-
a.x),num[tot++]=a.x,num[tot++]=b.x;}
    double gety(double x){return a.y+k*(x-a.x);;}
}l[N];
ll water[N];
bool cmp(segment s1,segment s2){return s1.a.x<s2.a.x;}
vector<segment>v;
struct Node
{
    int nxt,to;
    Node(int nxt=0,int to=0):nxt(nxt),to(to){}
}Edges[N*2];
int head[N],cnt=0;
void addedge(int u,int v){Edges[++cnt]=Node(head[u],v),head[u]=cnt;}
void dfs(int u)
{
    for(int i=head[u];~i;i=Edges[i].nxt)
    {
        int v=Edges[i].to;
        dfs(v),water[u]+=water[v];
```

```
    }
}
int main()
{
    int n;
    while(~scanf("%d",&n))
    {
        memset(head,-1,sizeof(head));
        memset(water,0,sizeof(water));
        tot=0,cnt=0;
        for(int i=0;i<n;i++)l[i].init(),l[i].id=i;
        sort(num,num+tot);
        tot=unique(num,num+tot)-num;
        sort(l,l+n,cmp);
        vector<segment>().swap(v);
        for(int i=0,j=0;i<tot;i++)
        {
            while(j<n&&l[j].a.x<=num[i])v.pb(l[j++]);
            vector<segment>v1;
            int top=-1;double maxh=-1;
            for(int p=0;p<v.size();p++)
            {
                segment s=v[p];
if((s.a.x==num[i]&&s.a.y<s.b.y)||(s.b.x==num[i]&&s.b.y<s.a.y))
                {
                    int k=n;double maxy=-1;
                    for(int q=0;q<v.size();q++)
if(v[q].id!=s.id&&v[q].gety(num[i])<s.gety(num[i])&&v[q].gety(num[i])>maxy)
maxy=v[q].gety(num[i]),k=v[q].id;
                    addedge(k,s.id);
                }
                if(s.b.x!=num[i])
                {
                    v1.pb(s);
                    if(s.gety(num[i])>maxh)maxh=s.gety(num[i]),top=s.id;
                }
            }
            v1.swap(v);
            if(top!=-1)water[top]+=num[i+1]-num[i];
        }
        dfs(n);
        for(int i=0;i<n;i++)printf("%lld\n",water[i]);
    }
    return 0;
}
```

POJ 1177 & HDU 1828 Picture

经典扫描线，求多个矩形并的周长。在左端把 $[y_1,y_2]$ 加入线段树，右端删除。维护区间的段数和有效长度。横方向上要被计入答案的是当前扫描到的线与线之间距离×2×（线段树中的）段数，竖方向是加入每个操作前后线段树里有效长度之差的绝对值。注意细节问题比如同一条线上的操作要先增后删啦之类。

```
#include<cstdio>
#include<vector>
#include<algorithm>
#include<cstring>
#define pii pair<int,int>
#define ll long long
#define pb push_back
using namespace std;
const int N=5005;
const int M=2e4+10;
struct Node
{
    int le,re,num,len,cnt;//段数 有覆盖长度 覆盖次数
}t[M*4];
void update(int idx)
{
    if(t[idx].cnt)return;
    t[idx].le=t[idx<<1].le,t[idx].re=t[idx<<1|1].re;
    t[idx].len=t[idx<<1].len+t[idx<<1|1].len;
    t[idx].num=t[idx<<1].num+t[idx<<1|1].num-
(t[idx<<1].re&&t[idx<<1|1].le);
}
void add(int idx,int l,int r,int a,int b,int d)
{
    if(a<=l&&b>=r)
    {
        t[idx].cnt+=d;
        if(t[idx].cnt)
        {
            t[idx].num=t[idx].le=t[idx].re=1;
            t[idx].len=r-l+1;
        }
        else if(l==r)t[idx].num=t[idx].le=t[idx].re=t[idx].len=0;
        else update(idx);
        return;
    }
    int mid=(l+r)>>1;
    if(b<=mid)add(idx<<1,l,mid,a,b,d);
    else if(a>mid)add(idx<<1|1,mid+1,r,a,b,d);
    else add(idx<<1,l,mid,a,b,d),add(idx<<1|1,mid+1,r,a,b,d);
    update(idx);
}
```

```
struct Node1{int a,b1,b2,d;}op[N*2];
bool cmp(Node1 x,Node1 y){return (x.a==y.a)?x.d>y.d:x.a<y.a;}
int num[N*2];
int main()
{
    int n;
    while(~scanf("%d",&n))
    {
        int tot=0,cnt=0,res=0;
        for(int i=0;i<n;i++)
        {
            int a1,b1,a2,b2;
            scanf("%d%d%d%d",&a1,&b1,&a2,&b2);
            op[tot++]=Node1{a1,b1,b2-1,1};
            op[tot++]=Node1{a2,b1,b2-1,-1};
            num[cnt++]=a1,num[cnt++]=a2;
        }
        sort(op,op+tot,cmp);
        sort(num,num+cnt);
        cnt=unique(num,num+cnt)-num;
        for(int i=0,j=0;i<cnt;i++)
        {
            while(j<tot&&op[j].a<=num[i])
            {
                int rec=t[1].len;
                add(1,-10001,10000,op[j].b1,op[j].b2,op[j].d),j++;
                res+=abs(t[1].len-rec);
            }
            res+=t[1].num*2*(num[i+1]-num[i]);
        }
        printf("%d\n",res);
    }
    return 0;
}
```

POJ 3277 City Horizon

给若干建筑求面积并（其实还是矩形面积并）。

```
#include<cstdio>
#include<vector>
#include<algorithm>
#include<cstring>
#define pii pair<int,int>
#define ll long long
```

```
#define pb push_back
using namespace std;
const int N=40005;
int numx[N*2],cnt1,numy[N],cnt2;
struct Node1{int a,h,d;}op[N*2];
bool cmp(Node1 x,Node1 y){return (x.a==y.a)?x.d>y.d:x.a<y.a;}
struct Node2{int cnt;ll len;}t[N*4];
void update(int idx)
{
    if(t[idx].cnt)return;
    t[idx].len=t[idx<<1].len+t[idx<<1|1].len;
}
void add(int idx,int l,int r,int a,int b,int d)
{
    if(l>=a&&r<=b)
    {
        t[idx].cnt+=d;
        if(t[idx].cnt)t[idx].len=numy[r]-numy[l-1];
        else if(l==r)t[idx].len=0;
        else update(idx);
        return;
    }
    int mid=(l+r)>>1;
    if(a>mid)add(idx<<1|1,mid+1,r,a,b,d);
    else if(b<=mid)add(idx<<1,l,mid,a,b,d);
    else add(idx<<1,l,mid,a,b,d),add(idx<<1|1,mid+1,r,a,b,d);
    update(idx);
}
int main()
{
    int n;
    while(~scanf("%d",&n))
    {
        cnt1=cnt2=0;
        for(int i=0;i<n;i++)
        {
            int a,b,h;
            scanf("%d%d%d",&a,&b,&h);
            op[2*i]=Node1{a,h,1};
            op[2*i+1]=Node1{b,h,-1};
            numx[++cnt1]=a,numx[++cnt1]=b;
            numy[++cnt2]=h;
        }
        sort(numx+1,numx+1+cnt1),cnt1=unique(numx+1,numx+1+cnt1)-numx-1;;
        sort(numy+1,numy+1+cnt2),cnt2=unique(numy+1,numy+1+cnt2)-numy-1;;
        sort(op,op+2*n,cmp);
        ll res=0;
        for(int i=1,j=0;i<=cnt1;i++)
        {
            while(j<2*n&&op[j].a<=numx[i])
            {
```

```
                int h=lower_bound(numy+1,numy+1+cnt2,op[j].h)-numy;
                add(1,1,cnt2,1,h,op[j].d),j++;
            }
            res+=(numx[i+1]-numx[i])*t[1].len;
        }
        printf("%lld\n",res);
    }
    return 0;
}
```

POJ 2280 & UVA 1606 Amphiphilic Carbon Molecules

给二维平面上若干黑白点，要用一条直线划分，问一侧黑点加另一侧白点加线上点数目和的最大值。

明显选其中两点连线是可以取到最大值的。枚举固定一点，极角排序，扫过去的同时维护一个半平面的区间即可。（实现技巧：把某个颜色的点变换为关于定点对称的点可以只数一侧点数便于统计；排序时用atan2判断是否在 180° 内用叉积避免精度问题）

```
#include<cstdio>
#include<vector>
#include<algorithm>
#include<cstring>
#include<cmath>
#define pii pair<int,int>
#define ll long long
#define pb push_back
using namespace std;
const int N=1e4+5;
struct Node1{ll x,y;int r;}p[N];
struct Node2{ll x,y;double ang;}v[N];
bool cmp(Node2 a,Node2 b){return a.ang<b.ang;}
ll cross(Node2 a,Node2 b){return a.x*b.y-a.y*b.x;}
int main()
{
    int n;
    while(~scanf("%d",&n)&&n)
    {
        int res=0;
        for(int i=1;i<=n;i++)scanf("%lld%lld%d",&p[i].x,&p[i].y,&p[i].r);
        for(int i=1;i<=n;i++)
        {
            int num=0,tot=0;
            for(int j=1;j<=n;j++)
            {
```

```
                if(i==j)continue;
                if(p[j].r)v[tot++]={p[j].x-p[i].x,p[j].y-
p[i].y,atan2(p[j].y-p[i].y,p[j].x-p[i].x)};
                else v[tot++]={p[i].x-p[j].x,p[i].y-p[j].y,atan2(p[i].y-
p[j].y,p[i].x-p[j].x)};
            }
            sort(v,v+tot,cmp);
            for(int j=0,k=0;j<tot;j++)
            {
                while(k<j+tot&&cross(v[j],v[k%tot])>=0)k++;
                num=max(num,k-j);
            }
            res=max(res,num+1);
        }
        printf("%d\n",res);
    }
    return 0;
}
```

POJ 3004 Subway planning

二维平面上给出若干个点，规定每点到某条原点出发的射线的距离不超过 d 问最少多少条射线。

每个点都可以确定一个射线角度的区间，转化成区间覆盖问题（每个区间中至少要选一个点），贪心，因为是环形的，枚举每一个区间作为起始。（拓展阅读：三类贪心区间覆盖问题 BY Ra煞

啊啊啊啊迷惑 31 行到 39 行这个部分我一开始用的atan2就不行。看别人的代码比对之后锁定问题就这段改成asin才能过。。

```
#include<cstdio>
#include<vector>
#include<algorithm>
#include<cstring>
#include<cmath>
#define pii pair<int,int>
#define ll long long
#define pb push_back
#define INF 0x3f3f3f3f
using namespace std;
const int N=505;
const double pi=acos(-1.0);
const double eps=1e-6;
struct Node{double a1,a2;}q[N];
int dcmp(double x){return x<-eps?-1:(x>eps);}
bool cmp(Node x,Node y){return x.a1<y.a1;}
int main()
```

```
{

    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n,d,res=INF,tot=0;
        scanf("%d%d",&n,&d);
        for(int i=1;i<=n;i++)
        {
            int x,y;
            scanf("%d%d",&x,&y);
            double a;
            if(x==0){if(y>0)a=0.5*pi;else a=1.5*pi;}
            else if(y==0){if(x>0)a=0;else a=pi;}
            else
            {
                a=asin(abs(y)/sqrt(x*x+y*y));
                if(x<0&&y>0)a=pi-a;
                else if(x<0&&y<0)a+=pi;
                else if(x>0&&y<0)a=2*pi-a;
            }
            double da=asin(d*1.0/sqrt(x*x+y*y));
            if(dcmp(sqrt(x*x+y*y)-d)<=0)continue;
            q[++tot]=Node{a-da,a+da};
        }
        sort(q+1,q+1+tot,cmp);
        for(int i=1;i<=tot;i++)
        {
            double r=-INF;int num=0;
            for(int j=0,k=i;j<tot;j++,k=(k+1>tot)?1:k+1)
            {
                double tl=q[k].a1,tr=q[k].a2;
                if(k<i)tl+=2*pi,tr+=2*pi;
                if(dcmp(tl-r)>0){num++,r=q[k].a2;}
                else r=min(r,q[k].a2);
            }
            res=min(res,num);
        }
        printf("%d\n",res==INF?0:res);
    }
    return 0;
}
```