2016百度之星复赛

小明在旅游的路上看到了一条美丽的河，河上有许多船只，有的船只向左航行，有的船只向右航行。小明希望拍下这一美丽的风景，并且把尽可能多的船只都完整地拍到一张照片中。  
  
  
小明位于河的边上，并且可以在河边的任意位置进行拍照，照相机的视野恰好为90度角，只能以垂直于河边的方向进行拍照。河上的船只全都可看作是平行于河边的一条线段，跟河边的距离各不相同，有的正在向左移动，有的正在向右移动，但移动速度恰好都是一样的。小明可以等待恰当的时间让尽量多的船只都走进照相机的视野里，你不需要考虑船只之间会互相遮挡视野的情况。

对于船[x,y,z]，当x+z>=y-z的时候，可以在[y-z,x+z]这些位置观测到它，那么我们可以处理出来n条线段，我们要求同一垂线交过的线段的数量最多，由于船是可以移动的，那么我们可以固定同一方向的船不动，假设初始在x位置观测往右的船，在y位置观测到往左的船，只要x<=y，那么经过一段时间我们就可以在同一角度观察到这些船。

那么如何求同一垂线交过的线段的数量最多，我们把线段的左端点的值当做1，右端点当成-1，然后就是一个累计的过程，我们要先预处理出每一个端点向左走的船的最大值，最后O（n）扫一遍即可。

总复杂度nlogn

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <vector>

using namespace std;

typedef long long ll;

struct node

{

int p,zy,dir;

};

bool cmp(node a,node b)

{

if(a.p!=b.p)return a.p<b.p;

if(a.zy!=b.zy)return a.zy<b.zy;

return a.dir>b.dir;

}

vector<node> v;

ll ans[10000000];

int main()

{

//freopen("input.txt","r",stdin);

ios::sync\_with\_stdio(false);

cin.tie(0);

int T,n,x,y,z,d;

cin>>T;

for(int cas=1; cas<=T; cas++)

{

cin>>n;

v.clear();

for(int i=0; i<n; i++)

{

cin>>x>>y>>z>>d;

//左航行是-1

int l=y-z,r=x+z;

node t;

if(l<=r)//否则不能完整拍到一只船

{

t.p=l;t.zy=-1;t.dir=d;

v.push\_back(t);//左端点是-1

t.p=r;t.zy=1;

v.push\_back(t);

}

}

sort(v.begin(),v.end(),cmp);

ll now(0);

ans[v.size()]=0;

//向左航行和向右航行的船分开处理

//预处理出每一个端点向左走的船的最大值

for(int i=v.size()-1; i>=0; i--)

//处理向左走的船

{

if(v[i].dir==-1 && v[i].zy==1)//向左航行 右端点

now++;

ans[i]=max(ans[i+1],now);//不论是不是左端点这一步都要操作

//ans[i]维护的是i到n-1点能看到的向左航行的船只的数目的最大值

if(v[i].dir==-1 && v[i].zy==-1)//向左航行 左端点

now--;

}

now=0;

ll cnt=0;

for(int i=0; i<v.size(); i++)

{

if(v[i].dir==1)

{

if(v[i].zy==-1)

now++;

cnt=max(ans[i]+now,cnt); //相对航行的船一定有一个时刻会相遇,共同出现在一个视野里

if(v[i].zy==1)

now--;

}

}

cout<<"Case #"<<cas<<":\n"<<cnt<<endl;;

}

return 0;

}