关于极角排序：

在平面内取一个定点 O，叫极点，引一条射线 Ox，叫做极轴，再选定一个长度单位和角度的正方向（通常取逆时针方向）。

对于平面内任何一点 M，用ρ表示线段 OM 的长度（有时也用 r 表示），θ表示从 Ox 到 OM 的角度，ρ叫做点 M 的极径，θ叫做点 M 的极角，有序数对 (ρ,θ) 就叫点 M 的极坐标。

那么给定平面上的一些点，把它们按照一个选定的中心点排成顺（逆）时针。（一般都是逆时针，绕着原点或者其他点）

极角排序常用的四种方法：

在说四种方法之前，给出一会用到的函数和存储点的结构体

struct point//存储点   
{   
 double x,y;   
};

方法 1：利用 atan2（）函数按极角从小到大排序。

关于 atan2() 函数：在 C 语言的 math.h 或 C++ 中的 cmath 中有两个求反正切的函数 atan(double x) 与 atan2(double y,double x) 他们返回的值是弧度要转化为角度再自己处理下。

前者接受的是一个正切值（直线的斜率）得到夹角，但是由于正切的规律性本可以有两个角度的但它却只返回一个，因为 atan 的值域是从 - 90~90 也就是它只处理一四象限，所以一般不用它。

第二个 atan2(double y,double x) 其中 y 代表已知点的 Y 坐标，同理 x , 返回值是此点与远点连线与 x 轴正方向的夹角，这样它就可以处理四个象限的任意情况了，它的值域相应的也就是 - 180~180 了,如果将斜率从小到大排序的话，那么顺序为3->4->1->2象限，权值返回从

-pi~pi

bool cmp1(point a,point b)  
{  
 if(atan2(a.y,a.x)!=atan2(b.y,b.x))  
 return atan2(a.y,a.x)<atan2(b.y,b.x);  
 else return a.x<b.x;  
}

方法 2：利用叉积按极角从小到大排序。

叉积

已知两点坐标，通过叉积可以求得与原点所围成的三角形的有向面积。  
比如这两个点为a,b.  
1/2∗(a.x∗b.y−a.y∗b.x)即为该三角形面积，那么为什么说是有向面积呢，如果这个值是正的，说明b位于a的正方向，即逆时针方向(当然，这个角度小于π)，反之，如果这个面积是负的，说明b位于a的负方向，即顺时针方向。叉积 = 0 是指两向量平行（重合）。  
那么我们就可以通过叉积来求极角了

double cross(double x1,double y1,double x2,double y2)　//计算叉积

{

return (x1\*y2-x2\*y1);

}

double compare(point a,point b,point c)//计算极角

{

return cross((b.x-a.x),(b.y-a.y),(c.x-a.x),(c.y-a.y));

}

point c;//原点，在函数外面定义，节省时间

c.x=0;c.y=0;

bool cmp2(point a,point b)//计算叉积，如果叉积相等，按照x从小到大排序

{double tmp=compare(c,a,b);

if(tmp==0)return a.x<b.x;//这里有的时候可以用eps来判定是否等于0

else return tmp>0;

}

方法 3：先按象限从小到大排序 再按极角从小到大排序

int Quadrant(point a)　　//象限排序，注意包含四个坐标轴   
{   
 if(a.x>0&&a.y>=0) return 1;   
 if(a.x<=0&&a.y>0) return 2;   
 if(a.x<0&&a.y<=0) return 3;   
 if(a.x>=0&&a.y<0) return 4;   
}   
   
bool cmp3(point a,point b) //先按象限从小到大排序 再按极角从小到大排序   
{   
 if(Quadrant(a)==Quadrant(b))//返回值就是象限   
 return cmp1(a,b);   
 else Quadrant(a)<Quadrant(b);   
}

关于三种方法的比较：

第三种方法按象限从小到大排序 再按极角从小到大排序是在有特殊需求的时候才会用到，这里不做比较。

关于第一种方法，利用 atan2 排序，他和利用叉积排序的主要区别在精度和时间上。

具体对比：时间：相较于计算叉积，利用 atan2 时间快，这个时间会快一点（记得做过一个题用 atan2 排序过了，用叉积的 T 了）

精度： atan2 精度不如叉积高，做过一个题用 anat2 因为精度问题 WA 了。

不过精度问题的话，可以把斜率用long double进行表示，atan2l (long double y,long double x）

就可以返回一个long double的值

所以两种方法根据情况选择一种合适的使用。

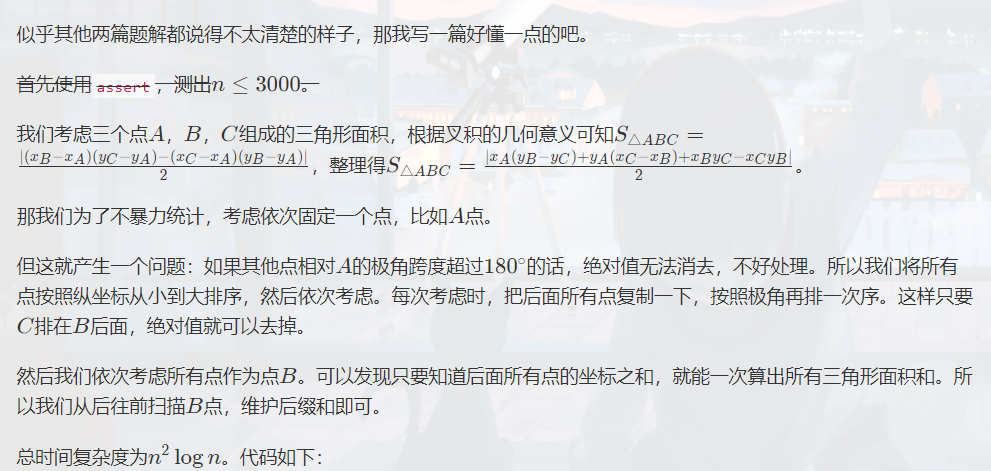
例题

1.[Luogu 2992](https://www.luogu.org/problemnew/show/P2992)  
题目大意：给你平面上n个点，保证任意两个点的连线都不经过原点，我们称所有包括原点的三角形为黄金三角形，问有多少个黄金三角形。  
n≤100000  
直接求黄金三角形好像有点难，我们考虑算出所有三角形的数量去减去非黄金三角形的数量。  
所有三角形的数量组合数算出。  
对于非黄金三角形，先确定一个点a，如果另外两个点均在我正方向π内，那么这三个点一定能构成一个非黄金三角形，如果不能理解的话可以画个图。  
我们先将所有的点极角排序，那么就可以一个指针扫过去知道有多少个点是在这个范围内的，组合数计算即可。

#include<cstdio>   
#include<algorithm>   
#include<cctype>   
#include<cstring>   
#include<iostream>   
#include<cmath>   
#define LL long long   
#define N (100005)   
using namespace std;   
int n,tot,r;   
LL ans;   
struct node{   
 int x,y;   
 long double angle;   
}a[N];   
inline bool cmp(node a,node b){   
 return a.angle<b.angle;   
}   
template <typename T> void read(T&t) {   
 t=0;   
 bool fl=true;   
 char p=getchar();   
 while (!isdigit(p)) {   
 if (p=='-') fl=false;   
 p=getchar();   
 }   
 do {   
 (t\*=10)+=p-48;p=getchar();   
 }while (isdigit(p));   
 if (!fl) t=-t;   
}   
inline bool cj(int x,int y){   
 return 1ll\*a[x].x\*a[y].y>=1ll\*a[x].y\*a[y].x;//逆时针旋转小于pi   
}   
int main(){   
 read(n);   
 for (int i=1;i<=n;i++) read(a[i].x),read(a[i].y),a[i].angle=atan2(a[i].y,a[i].x);   
 sort(a+1,a+n+1,cmp);   
 r=2;   
 for (int i=1;i<=n;i++){   
 while (r!=i&&cj(i,r)) tot++,r=r%n+1;   
 ans-=1ll\*tot\*(tot-1)/2;   
 tot--;   
 }   
 ans+=1ll\*n\*(n-1)\*(n-2)/6;   
 printf("%lld",ans);   
 return 0;   
}

2.[Luogu3476](https://www.luogu.org/problemnew/show/P3476#sub)

题目大意：给定平面上n个位于第一象限的点，求这些点所能组成的三角形的面积和。  
0≤n≤3000

  
#define LL long long   
#define N (3005)   
using namespace std;   
template <typename T> void read(T&t) {   
 t=0;   
 bool fl=true;   
 char p=getchar();   
 while (!isdigit(p)) {   
 if (p=='-') fl=false;   
 p=getchar();   
 }   
 do {   
 (t\*=10)+=p-48;p=getchar();   
 }while (isdigit(p));   
 if (!fl) t=-t;   
}   
int n,tot;   
LL sumx,sumy,ans;   
struct node{   
 int x,y;   
}a[N],q[N];   
inline bool cmp(node a,node b){   
 return a.x==b.x?a.y<b.y:a.x<b.x;   
}   
inline bool cmp1(node a,node b){   
 return 1ll\*a.x\*b.y-1ll\*a.y\*b.x>0;   
}   
int main(){   
 read(n);   
 for (int i=1;i<=n;i++){   
 read(a[i].x),read(a[i].y);   
 }   
 sort(a+1,a+n+1,cmp);   
 for (int i=1;i<=n;i++){   
 tot=0;   
 for (int j=i+1;j<=n;j++){   
 tot++;   
 q[tot].x=a[j].x-a[i].x;   
 q[tot].y=a[j].y-a[i].y;   
 }   
 sort(q+1,q+tot+1,cmp1);   
 sumx=0,sumy=0;   
 for (int j=1;j<=tot;j++) sumx+=q[j].x,sumy+=q[j].y;   
 for (int j=1;j<=tot;j++){   
 sumx-=q[j].x,sumy-=q[j].y;   
 ans+=sumy\*q[j].x-sumx\*q[j].y;   
 }   
 }   
 printf("%lld.%lld",ans/2,(ans&1)\*5);   
 return 0;   
}