C5_C

“这是一道凸包模板题”

老说常说：“世上无难题，只要有模板”（bushi）

一看标题，就知道~~如果出题人没有恶意~~，这就是一道标准的模板题（ctrl C-V题）

悄悄瞥了一眼数据大小：

（1≤t≤50）（3≤n≤10^3）（0≤xi,yi≤10^9）

关于时间：MAX=50*3*10^3≈10^5＜＜1000ms

关于大小：10^9过大，使用int可能出问题（且确实出了），
考虑多使用double或者long long

让我们来愉快的
找模板（答案）吧！

第一步：翻PPT

```
GRAHAM-SCAN(Q)
1  设p0是Q中y坐标最小的点，或者是最左边的点
2  设<p1, p2，…， pm>是Q中剩下的点，围绕p0按极角逆
   时针顺序排序(如果有多个点具有相同的角度，留下离
   p0最远的那个点)
3  PUSH(p0, S)
4  PUSH(p1, S)
5  PUSH(p2, S)
6  for i ← 3 to m
7      while ( 由NEXT-TO-TOP(S)、TOP(S)和pi组成
             的连续线段不向左转弯 ) // 直线，栈顶点在
             凸包边上；右转，点在凸包内
8          POP(S)
9      PUSH(pi, S) // pi 可能为凸包顶点，入栈
10 return S
```

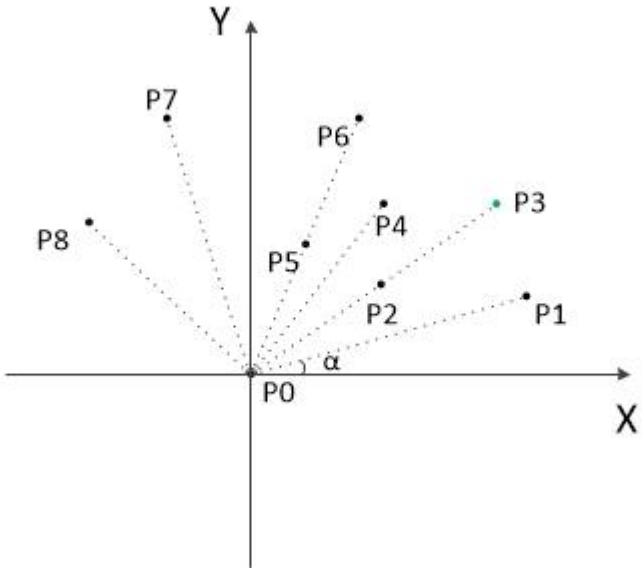
无敌巨佬：这还用看？

我的大佬同学：考虑如何将伪代码转化成为代码

菜狗~~且不上课听讲~~的我：这啥呀！！

第二步：了解Graham's scan算法

Graham扫描的思想是先找到凸包上的一个点，然后从那个点开始按逆时针方向逐个找凸包上的点，实际上就是进行极角排序，然后对其查询使用。



（可莉明白了！就是PPT里的1，2步）

秉持着学一步有进一步的收获，先实现极角排序的代码

1.输入并寻找左下角的点

```
for(int i=0;i<n;i++){
            scanf(" %lf %lf",&points[i].x,&points[i].y);
            if(points[the_point].y>points[i].y){
                the_point=i;
             }
            else if(points[the_point].y==points[i].y){
                if(points[the_point].x>points[i].x){
                    the_point=i;
                }
            }
        }              //输入并寻找the_point(坐标原点);
```

2.鉴于所有点都在the_point的上方，选择用余弦值排序

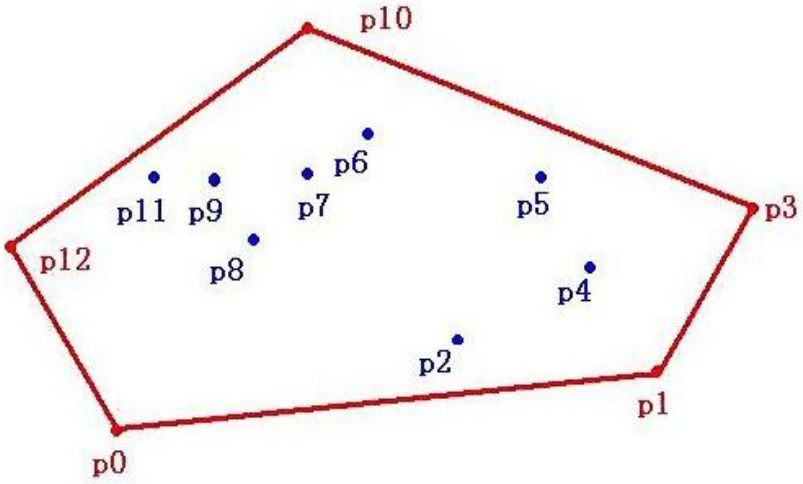
```
for(int i=0;i<n;i++){
            if(points[i].y==points[the_point].y){
                if(points[i].x>points[the_point].x){
                    points[i].cos=1;
                }
                else{
                    points[i].cos=2;
                }
            }
            else{
                points[i].dist=disT(points[i],points[the_point]);
                points[i].cos=(points[i].x-points[the_point].x)/(points[i].dist);
            }
        }       //  求出每一个点相对于the_point的距离与余弦值;
sort(points,points+n,compare);  //根据余弦值排序
```

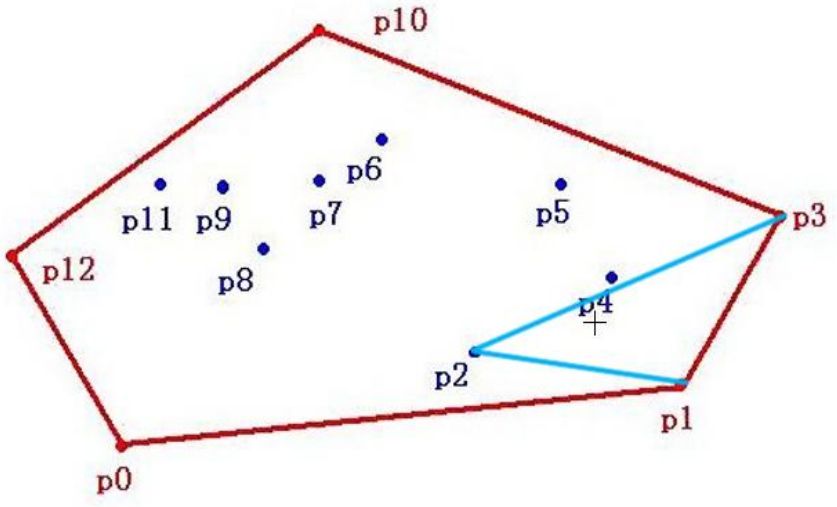
```
double disT(point a,point b){
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
```

```
bool compare(const point& a,const point& b){
    if(a.cos>b.cos){
        return true;
     }
    else{
        if(a.cos==b.cos){
            if(a.dist<b.dist){
                return true;
            }
        }
        else{
            return false;
        }
    }
}       //sort中的结构体排序
```

第三步：如何“查询使用”？

通过保持候选点的栈S，令连续线段向左或向右转





（显然P1，P2，P3转折是线段向右）

看了一眼闭包图，思考了一下如果把P2加进去后.....！明白了什么叫“连续线段向左”

代码实现：

```
3  PUSH(p0, S)
4  PUSH(p1, S)
5  PUSH(p2, S)
6  for i ← 3 to m
7      while ( 由NEXT-TO-TOP(S)、TOP(S)和pi组成
                的连续线段不向左转弯 ) // 直线，栈顶点在
                凸包边上；右转，点在凸包内
8          POP(S)
9      PUSH(pi, S) // pi 可能为凸包顶点，入栈
10 return S
```

显然，S是栈类型，可以用C++中的stack，但是伪代码中的NEXT-TO-TOP(S)用stack却不便于实现，鉴于数据中n不大，可以直接采用数组:ans[]+int数:ans_num充当栈

```
int ans[1005],ans_num=0,lines=0;
        ans[ans_num++]=0;
        ans[ans_num++]=1;
        ans[ans_num]=2;
        for(int i=3;i<n;i++){
            while(F_judge(points[ans_num-1],points[ans_num],points[i])<=0){
                ans[ans_num--]=0;
            }
            ans_num++;
            ans[ans_num]=i;
        }                   //求得闭包上的点组ans;
```

第四步：如何实现F_judge?

向量积

设点a(x1,y1)
设点b(x2,y2)
axb=|a|*|b|*sinθ=x1*y2-x2*y1

根据向量积的公式，我们可以得出:

axb=5x3-3x0=15>0

bxa=3x0-5x3=-15<0

在这里我们直接给出结论:

如果axb为正数，

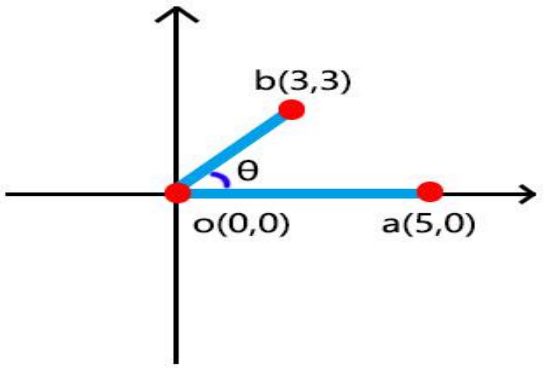
则b线在a线的逆时针方向(围绕圆心);

如果axb为负数，

则b线在a线的顺时针方向;

如果axb为0，

则a线和b线是重合的。



代码实现：

```
double F_judge(point a,point b,point c){
  return (b.x-a.x)*(c.y-b.y)-(c.x-b.x)*(b.y-a.y);
}                    //根据向量机判断边是否“右转”;
```

第五步：已知凸包上的点求周长

```
for(int i=0;i<ans_num;i++){
    point_1=points[ans[i]];
    point_2=points[ans[i+1]];
    C_part[lines++]=disT(point_1,point_2);
}       //根据ans求各个边长
C_part[lines]=disT(points[ans[ans_num]],points[0]);      //最后一条边
for(int i=0;i<=lines;i++){
    C_all=C_all+C_part[i];
}                //求和
printf("%.10f\n",C_all);
```

（总算是有我会的了）

AC代码（封装版）：

```
#include<iostream>
#include<algorithm>
#include<cmath>
#include<cstdlib>
using namespace std;
struct point{
    double x;
    double y;
    double cos;
    double dist;
}points[1005],point_1,point_2;
int the_point,lines,n,ans[1005],ans_num;
double C_part[1005],C_all;
```

```
bool compare(const point& a,const point& b){
    if(a.cos>b.cos){
        return true;
    }
    else{
        if(a.cos==b.cos){
            if(a.dist<b.dist){
                return true;
            }
        }
        else{
            return false;
        }
    }
}       //sort中的结构体排序
double F_judge(point a,point b,point c){
    return (b.x-a.x)*(c.y-b.y)-(c.x-b.x)*(b.y-a.y);
}               //根据向量机判断边是否“右转”;
double disT(point a,point b){
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
```

```
void input(){
    for(int i=0;i<n;i++){
            scanf(" %lf %lf",&points[i].x,&points[i].y);
            if(points[the_point].y>points[i].y){
                the_point=i;
            }
            else if(points[the_point].y==points[i].y){
                if(points[the_point].x>points[i].x){
                    the_point=i;
                }
            }
        }               //输入并寻找the_point(坐标原点);
}
```

```
void refine_point(){
    for(int i=0;i<n;i++){
        if(points[i].y==points[the_point].y){
            if(points[i].x>points[the_point].x){
                points[i].cos=1;
            }
            else{
                points[i].cos=2;
            }
        }
        else{
            points[i].dist=disT(points[i],points[the_point]);
            points[i].cos=(points[i].x-points[the_point].x)/(points[i].dist);
        }
    }       //  求出每一个点相对于the_point的距离与余弦值;
}
```

```
void get_ans(){
    ans_num=0;
    lines=0;
    ans[ans_num++]=0;
    ans[ans_num++]=1;
    ans[ans_num]=2;
    for(int i=3;i<n;i++){
        while(F_judge(points[ans_num-1],points[ans_num],points[i])<=0){
            ans[ans_num--]=0;
        }
        ans_num++;
        ans[ans_num]=i;
    }               //求得闭包上的点组ans；
}
```

```
void ans_to_C(){
    for(int i=0;i<ans_num;i++){
        point_1=points[ans[i]];
        point_2=points[ans[i+1]];
        C_part[lines++]=disT(point_1,point_2);
    }       //根据ans求各个边长
    C_part[lines]=disT(points[ans[ans_num]],points[0]);     //最后一条边
    for(int i=0;i<=lines;i++){
        C_all=C_all+C_part[i];
    }               //求和
}
```

```cpp
int main(){
    int t;
    cin >> t;
    for(int k=0;k<t;k++){
        scanf("%d",&n);
        input();
        refine_point();
        sort(points,points+n,compare);   //根据余弦值排序
        get_ans();
        ans_to_C();
        printf("%.10f\n",C_all);
    }
    return 0;
}
```

开开心心AC啦！
OS:（希望助教下一次还出板子）

谢谢大家观看!祝大家次次~~及格~~AK!