E5B 旋转卡壳怎么读

题目描述

二维平面上有 n 个点，坐标分别为 (x1,y1),(x2,y2),...,(xn,yn)，要求出任意两点间最大欧几里得距离，即：

$$\max_{1\le i<j\le n}\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$$

输入格式

第一行一个正整数 n（2≤n≤10^5），表示点的数量。

接下来 n 行，每行两个至多六位小数的浮点数 xi,yi（0≤|xi|,|yi|≤5×10^5），表示第 i 个点的坐标。

输出格式

输出一行一个浮点数，保留六位小数，表示任意两点间最大欧几里得距离。

输入样例

```
4
0 1
1 0
1 1
-1 -1
```

输出样例

```
1
2
```

题解思路

要寻找n个点中距离最远的两个点，我们可以知道这两个点一定位于点集的最外圈，即它的凸包上，最远距离就是凸包的直径，为了得到这两个点，我们需要先求出点集的凸包，再利用旋转卡壳算法找到凸包上距离最大的点对，输出其欧几里得距离。

首先求凸包：找到最左下角的点作为原点，将所有点按照极角大小排序，之后通过判断叉乘的正负来确定是否为逆时针旋转，最后将凸包中的点放入一个栈中。求凸包的过程与C5-C一致。（此处需要注意本题数据有全部共线情况，对于这种情况要注意在按照极角排序的比较函数中加入极角相等时的处理，即按照距离大小排序，距离原点小的在前。）

之后旋转卡壳：遍历凸包中的点，对于每个点，通过计算叉乘的绝对值找到与其构成三角形面积最大的线段，记录该点与线段两端点中距离较大者的距离，对于每个凸包中的点，记录并更新最大距离，最后得到凸包的直径并输出，即点集中任意两点的最大距离。（当寻找最大面积时，面积是先增大后减小的，只需找到单调性改变的位置即可。对于下一个凸包中的点，只需从上一个点所对应的最大面积线段

处开始找起。）

代码

```
#include<iostream>
#include<algorithm>
#include<cstdio>
#include<cmath>
using namespace std;
long long n;
struct node{
    double x,y;
}p[100105],s[100105];
double check(node a1,node a2,node b1,node b2){
    return (a2.x-a1.x)*(b2.y-b1.y)-(b2.x-b1.x)*(a2.y-a1.y);
}
double d(node p1,node p2){
    return sqrt((p2.y-p1.y)*(p2.y-p1.y)+(p2.x-p1.x)*(p2.x-p1.x));
}
bool cmp(node p1,node p2){
    double tmp=check(p[1],p1,p[1],p2);
    if(tmp>1e-7)
        return 1;
    else if((fabs(tmp)<1e-7)&&d(p[1],p1)<d(p[1],p2))
        return 1;
    else
        return 0;
}
int main(){
    long long times;
    double ans=0,temp=0;
    //p和s都是从1开始
    scanf("%lld",&n);
    double mid;
    for(long long i=1;i<=n;i++){
        scanf("%lf%lf",&p[i].x,&p[i].y);
        if(i!=1&&((p[i].y<p[1].y)||(p[i].y==p[1].y && p[i].x<p[1].x))){
            mid=p[1].y;p[1].y=p[i].y;p[i].y=mid;
            mid=p[1].x;p[1].x=p[i].x;p[i].x=mid;
        }
    }
    sort(p+2,p+1+n,cmp);
    s[1]=p[1];
    long long cnt=1;
    for(long long i=2;i<=n;i++){
        while(cnt>1&&(!(check(s[cnt-1],s[cnt],s[cnt],p[i])>1e-10))){
            cnt--;
        }
        cnt++;
        s[cnt]=p[i];
    }
    s[cnt+1]=p[1];
    //旋转卡壳
    long long j=2;
    for(long long i=1;i<=cnt;i++){
```

```
        while(fabs(check(s[j],s[i],s[j],s[i+1]))
<fabs(check(s[j+1],s[i],s[j+1],s[i+1]))){
            j++;
            if(j>cnt)
                j=1;
        }
        temp=max(d(s[i],s[j]), d(s[i+1],s[j]));
        if(temp>ans)
            ans=temp;
    }
    printf("%.6lf",ans);
    return 0;
}
```