题目分析

- 这是一个旋转卡壳的模板题
- 使用`GrahamScan`找到凸包，然后对凸包上的点，寻找距离最大的两个点之间的距离，寻找方法如下

```
double Rotating_Calipers(int n, Point* P) {
    int b = 2;
    double ans = 0;
    P[n+1] = P[1];
    for(int a = 1; a <= n; a++) {
        while(abs(Xmult(P[a], P[a+1], P[b])) < abs(Xmult(P[a], P[a+1],
                P[b+1])))
            b = (b + 1) % n;
        ans = max(ans, max(Dist(P[a], P[b]), Dist(P[a+1], P[b])));
    }
    return ans;
}
```

代码

```
#include<bits/stdc++.h>
#define maxn 100010
#define eps (1e-20)
#define inf 0x3f3f3f3f
using namespace std;
int top;
typedef struct {
    double x, y;
} Point;
Point point[maxn], S[maxn];
int sgn(double x) {
    if(fabs(x) < eps) return 0;
    if(x < 0) return -1;
    return 1;
}
double Xmult(Point p0, Point p1, Point p2) { //p0为起点，返回p0p1与p0p2的叉积
    return ((p1.x-p0.x) * (p2.y-p0.y) - (p1.y-p0.y) * (p2.x-p0.x));
}
double Dist(Point p1, Point p2) {
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
bool cmp(Point a,Point b) {
    double k = Xmult(point[1], a, b);
// 极角大小排序，若共线，则距离远的排在前
    if(sgn(k) > 0 || (sgn(k) == 0 && sgn(Dist(point[1],a) -
                                         Dist(point[1],b)) > 0))
        return true;
    return false;
```

```
}
void GrahamScan(int n) {
    int u = 1; //记为1则点是从point数组的1开始记入
    for(int i = 2; i <= n; i++) //找到最左下的点p0
        if((point[i].y < point[u].y) || (point[i].y == point[u].y &&
                                         point[i].x < point[u].x))
            u = i;
    swap(point[u], point[1]);
    sort(point+2, point+n+1, cmp); //根据极角排序
    S[1] = point[1];
    S[2] = point[2];
    S[3] = point[3];
    top = 3;
    for(int i = 4; i <= n; i++) {
        while(Xmult(S[top-1], S[top], point[i]) < 0 && top > 0) //发生右转或共线就删
除栈顶点
            top--;
        S[++top] = point[i];
    }
}
double Rotating_Calipers(int n, Point* P) {
    int b = 2;
    double ans = 0;
    P[n+1] = P[1];
    for(int a = 1; a <= n; a++) {
        while(abs(Xmult(P[a], P[a+1], P[b])) < abs(Xmult(P[a], P[a+1],
                P[b+1])))
            b = (b + 1) % n;
        ans = max(ans, max(Dist(P[a], P[b]), Dist(P[a+1], P[b])));
    }
    return ans;
}
int main() {
    int n;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
        scanf("%lf %lf",&point[i].x, &point[i].y);
    GrahamScan(n);
    printf("%.6f", Rotating_Calipers(top, S));
    return 0;
}
```