半平面交 - CSDN博客

首先解决问题：**什么是半平面？**顾名思义，半平面就是指平面的一半，我们知道，一条直线可以将平面分为两个部分，那么这两个部分就叫做两个半平面。

然后，**半平面怎么表示呢？**二维坐标系下，直线可以表示为ax + by + c = 0，那么两个半平面则可以表示为ax + by + c >= 0 和ax + by + c < 0，这就是半平面的表示方法。

还有，**半平面的交是神马玩意？** 其实就是一个方程组，让你画出满足若干个式子的坐标系上的区域（类似于线性规划的可行域），方程组就是由类似于上面的这些不等式组成的。

另外，**半平面交可以干什么？** 半平面交虽然说是半平面的问题，但它其实就是关于直线的问题。一个一个的半平面其实就是一个一个有方向的直线而已。

半平面交的一个重要应用就是**求多边形的核**。 多边形的核又是神马玩意？ 它是平面简单多边形的核是该多边形内部的一个点集，该点集中任意一点与多边形边界上一点的连线都处于这个多边形内部。就是一个在一个房子里面放一个摄像 头，能将所有的地方监视到的放摄像头的地点的集合即为多边形的核。经常会遇到让你判定一个多边形是否有核的问题。

比如说：

POJ 3335 Rotating Scoreboard

http://acm.pku.edu.cn/JudgeOnline/problem?id=3335

POJ 1474 Video Surveillance

http://acm.pku.edu.cn/JudgeOnline/problem?id=1474

POJ 1279 Art Gallery

http://acm.pku.edu.cn/JudgeOnline/problem?id=1279

这三个题比较简单，裸的半平面交，下面是核心代码（暨半平面交的模板）：

[cpp]view plaincopy

```cpp
/*半平面相交（直线切割多边形）（点标号从1开始）*/
Point points[MAXN],p[MAXN],q[MAXN];
int n;
double r;
int cCnt,curCnt;
inlinevoid getline(Point x,Point y,double &a,double &b,double &c){
    a = y.y - x.y;
    b = x.x - y.x;
    c = y.x * x.y - x.x * y.y;
}
inlinevoid initial(){
for(int i = 1; i <= n; ++i)p[i] = points[i];
    p[n+1] = p[1];
    p[0] = p[n];
    cCnt = n;
}
inline Point intersect(Point x,Point y,double a,double b,double c){
double u = fabs(a * x.x + b * x.y + c);
double v = fabs(a * y.x + b * y.y + c);
return Point( (x.x * v + y.x * u) / (u + v) , (x.y * v + y.y * u) / (u + v) );
}
inlinevoid cut(double a,double b ,double c){
    curCnt = 0;
for(int i = 1; i <= cCnt; ++i){
if(a*p[i].x + b*p[i].y + c >= EPS)q[++curCnt] = p[i];
else {
if(a*p[i-1].x + b*p[i-1].y + c > EPS){
                q[++curCnt] = intersect(p[i],p[i-1],a,b,c);
            }
if(a*p[i+1].x + b*p[i+1].y + c > EPS){
                q[++curCnt] = intersect(p[i],p[i+1],a,b,c);
            }
        }
    }
for(int i = 1; i <= curCnt; ++i)p[i] = q[i];
```

```
    p[curCnt+1] = q[1];p[0] = p[curCnt];
    cCnt = curCnt;
}
inlinevoid solve(){
//注意：默认点是顺时针，如果题目不是顺时针，规整化方向
    initial();
for(int i = 1; i <= n; ++i){
double a,b,c;
        getline(points[i],points[i+1],a,b,c);
        cut(a,b,c);
    }
/*
    如果要向内推进r，用该部分代替上个函数
    for(int i = 1; i <= n; ++i){
        Point ta, tb, tt;
        tt.x = points[i+1].y - points[i].y;
        tt.y = points[i].x - points[i+1].x;
        double k = r / sqrt(tt.x * tt.x + tt.y * tt.y);
        tt.x = tt.x * k;
        tt.y = tt.y * k;
        ta.x = points[i].x + tt.x;
        ta.y = points[i].y + tt.y;
        tb.x = points[i+1].x + tt.x;
        tb.y = points[i+1].y + tt.y;
        double a,b,c;
        getline(ta,tb,a,b,c);
        cut(a,b,c);
    }
    */
//多边形核的面积
double area = 0;
for(int i = 1; i <= curCnt; ++i)
        area += p[i].x * p[i + 1].y - p[i + 1].x * p[i].y;
    area = fabs(area / 2.0);
//此时cCnt为最终切割得到的多边形的顶点数，p为存放顶点的数组
}
inlinevoid GuiZhengHua(){
//规整化方向，逆时针变顺时针，顺时针变逆时针
for(int i = 1; i < (n+1)/2; i ++)
      swap(points[i], points[n-i]);//头文件加iostream
}
inlinevoid init(){
for(int i = 1; i <= n; ++i)points[i].input();
        points[n+1] = points[1];
}
```

半平面交的最终奥义就是**求出一个满足条件的凸多边形**，而解决这个问题的前提就是**直线切割多边形**，让直线不断的去切割当前多边形，然后得到新的多边形，然后继续。。。最终得到一个符合条件的多边形，如果最终的多边形顶点数目为0，那么就说明半平面交无解（半平面交的解可以是一个凸多边形，一条直线，一个点，我们用顶点来记录这些解）。关于直线切割多边形，**流程**是这样的：

对 凸多边形（指的是当前多边形）的每一个顶点，如果这个顶点在直线的指定的一侧（暨在该半平面上），那么就将该顶点直接加入到新的多边形中，否则，看与该点 相邻的多边形上的两个点（判断线段是否和直线相交），如果和直线相交，则把交点加入到新的多边形中。这样，就可以得到一个新的凸多边形。关于最初的多边 形，我们可以设一个四方形的区域，比如说（-1000，-1000），（-1000,1000），（1000,1000），（1000,-1000），然 后开始切割。

POJ 3525 Most Distant Point from the Sea （推荐）

http://acm.pku.edu.cn/JudgeOnline/problem?id=3525

求在多边形中画一个圆的最大半径

可以使用半平面交+ 二分法解。对每个距离进行判定（是否存在区域可以放置圆心，多边形的的每条边向内推进

[cpp]view plaincopy

```
#include
#include
#include
#include
```

```
5. usingnamespace std;
6. #define EPS 1e-10
7. constint MAXN = 200;
8. struct Point{
9. double x,y;
10.     Point(){}
11.     Point(double _x,double _y):x(_x),y(_y){}
12. void input(){
13.         scanf("%lf%lf",&x,&y);
14.     }
15. };
16. Point points[MAXN],p[MAXN],q[MAXN];
17. int n;
18. int cCnt,curCnt;
19. inlinevoid getline(Point x,Point y,double &a,double &b,double &c){
20.      a = y.y - x.y;
21.     b = x.x - y.x;
22.     c = y.x * x.y - x.x * y.y;
23. }
24. inlinevoid initial(){
25. for(int i = 1; i <= n; ++i)p[i] = points[i];
26.     p[n+1] = p[1];
27.     p[0] = p[n];
28.     cCnt = n;
29. }
30. inline Point intersect(Point x,Point y,double a,double b,double c){
31. double u = fabs(a * x.x + b * x.y + c);
32. double v = fabs(a * y.x + b * y.y + c);
33. return Point( (x.x * v + y.x * u) / (u + v) , (x.y * v + y.y * u) / (u + v) );
34. }
35. inlinevoid cut(double a,double b ,double c){
36.     curCnt = 0;
37. for(int i = 1; i <= cCnt; ++i){
38. if(a*p[i].x + b*p[i].y + c >= 0)q[++curCnt] = p[i];
39. else {
40. if(a*p[i-1].x + b*p[i-1].y + c > 0){
41.                 q[++curCnt] = intersect(p[i],p[i-1],a,b,c);
42.             }
43. if(a*p[i+1].x + b*p[i+1].y + c > 0){
44.                 q[++curCnt] = intersect(p[i],p[i+1],a,b,c);
45.             }
46.         }
47.     }
48. for(int i = 1; i <= curCnt; ++i)p[i] = q[i];
49.     p[curCnt+1] = q[1];p[0] = p[curCnt];
50.     cCnt = curCnt;
51. }
52. inlineint solve(double r){
53.     initial();
54. for(int i = 1; i <= n; ++i){
55.         Point ta, tb, tt;
56.         tt.x = points[i+1].y - points[i].y;
57.         tt.y = points[i].x - points[i+1].x;
58. double k = r / sqrt(tt.x * tt.x + tt.y * tt.y);
59.         tt.x = tt.x * k;
60.         tt.y = tt.y * k;
61.         ta.x = points[i].x + tt.x;
62.         ta.y = points[i].y + tt.y;
63.         tb.x = points[i+1].x + tt.x;
64.         tb.y = points[i+1].y + tt.y;
65. double a,b,c;
66.         getline(ta,tb,a,b,c);
67.         cut(a,b,c);
68.     }
69. if(cCnt <= 0)return 0;
```

```
70. return 1;
71. }
72. int main(){
73. while(scanf("%d",&n) != EOF){
74. if(n == 0)break;
75. for(int i = 1; i <= n; ++i)points[i].input();
76. for(int i = 1; i < (n+1)/2; i ++)
77.             swap(points[i], points[n-i]);
78.         points[n+1] = points[1];
79. double high = 10000000;
80. double low = 0,mid;
81. while(low + EPS <= high){
82.             mid = (high + low)/2.0;
83. if(solve(mid))low = mid;
84. else high = mid;
85.         }
86.         printf("%lf/n",high);
87.     }
88. return 0;
89. }
```

POJ 3384 Feng Shui （推荐）

http://acm.pku.edu.cn/JudgeOnline/problem?id=3384

半平面交实际应用，用两个圆覆盖一个多边形，问最多能覆盖多边形的面积。

解法：用半平面交将多边形的每条边一起向“内”推进R，得到新的多边形，然后求多边形的最远两点

[cpp]view plaincopy

```
1. #include
2. #include
3. #include
4. #include
5. usingnamespace std;
6. #define EPS 1e-10
7. constint MAXN = 300;
8. struct Point{
9. double x,y;
10.     Point(){}
11.     Point(double _x,double _y):x(_x),y(_y){}
12. void input(){
13.         scanf("%lf%lf",&x,&y);
14.     }
15. };
16. Point points[MAXN],p[MAXN],q[MAXN];
17. int n;
18. double r;
19. int cCnt,curCnt;
20. void getline(Point x,Point y,double &a,double &b,double &c){
21.  a = y.y - x.y;
22.  b = x.x - y.x;
23.  c = y.x * x.y - x.x * y.y;
24. }
25. inlinevoid initial(){
26. for(int i = 1; i <= n; ++i)p[i] = points[i];
27.     p[n+1] = p[1];
28.     p[0] = p[n];
29.     cCnt = n;
30. }
31. inline Point intersect(Point x,Point y,double a,double b,double c){
32. double u = fabs(a * x.x + b * x.y + c);
33. double v = fabs(a * y.x + b * y.y + c);
34. return Point( (x.x * v + y.x * u) / (u + v) , (x.y * v + y.y * u) / (u + v) );
35. }
36. inlinevoid cut(double a,double b ,double c){
37.     curCnt = 0;
```

```
38. for(int i = 1; i <= cCnt; ++i){
39. if(a*p[i].x + b*p[i].y + c >= 0)q[++curCnt] = p[i];
40. else {
41. if(a*p[i-1].x + b*p[i-1].y + c > 0){
42.                 q[++curCnt] = intersect(p[i],p[i-1],a,b,c);
43.             }
44. if(a*p[i+1].x + b*p[i+1].y + c > 0){
45.                 q[++curCnt] = intersect(p[i],p[i+1],a,b,c);
46.             }
47.         }
48.     }
49. for(int i = 1; i <= curCnt; ++i)p[i] = q[i];
50.     p[curCnt+1] = q[1];p[0] = p[curCnt];
51.     cCnt = curCnt;
52. }
53. inlinedouble pdis(Point a,Point b){
54. return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
55. }
56. int main(){
57.     scanf("%d%lf",&n,&r);
58. for(int i = 1; i <= n; ++i)points[i].input();
59.     points[n+1] = points[1];
60.     initial();
61. for(int i = 1; i <= n; ++i){
62.         Point ta, tb, tt;
63.         tt.x = points[i+1].y - points[i].y;
64.         tt.y = points[i].x - points[i+1].x;
65. double k = r / sqrt(tt.x * tt.x + tt.y * tt.y);
66.         tt.x = tt.x * k;
67.         tt.y = tt.y * k;
68.         ta.x = points[i].x + tt.x;
69.         ta.y = points[i].y + tt.y;
70.         tb.x = points[i+1].x + tt.x;
71.         tb.y = points[i+1].y + tt.y;
72. double a,b,c;
73.         getline(ta,tb,a,b,c);
74.         cut(a,b,c);
75.     }
76. int ansx = 0,ansy = 0;
77. double res = 0;
78. for(int i = 1; i <= cCnt; ++i){
79. for(int j = i + 1; j <= cCnt; ++j){
80. double tmp = pdis(p[i],p[j]);
81. if(tmp > res){
82.                 res = tmp;
83.                 ansx = i;
84.                 ansy = j;
85.             }
86.         }
87.     }
88.     printf("%.4lf %.4lf %.4lf %.4lf/n",p[ansx].x,p[ansx].y,p[ansy].x,p[ansy].y);
89. return 0;
90. }
```

POJ 1755 Triathlon （推荐）

http://acm.pku.edu.cn/JudgeOnline/problem?id=1755

半平面交判断不等式是否有解。注意特殊情况

[cpp]view plaincopy

```
1. #include
2. #include
3. #include
4. usingnamespace std;
5. #define EPS 1e-8
6. constint MAXN = 110;
```

```
7. struct Point{
8. double x,y;
9.     Point(){}
10.     Point(double _x,double _y):x(_x),y(_y){}
11. };
12. struct Node{
13. double x,y,z;
14. void input(){
15.         scanf("%lf%lf%lf",&x,&y,&z);
16.     }
17. };
18. inlinevoid getabc(double &a,double &b,double &c,Node x1,Node x2){
19.     a = 1.0/x2.x - 1.0/x1.x ;
20.     b = 1.0/x2.y - 1.0/x1.y;
21.     c = 1.0/x2.z - 1.0/x1.z;
22. }
23. Point points[MAXN],p[MAXN],q[MAXN];
24. int n;
25. int cCnt,curCnt;
26. int flag ;
27. int sig(double k){
28. return (k < -EPS) ? -1 : (k > EPS);
29. }
30. inlinevoid getline(Point x,Point y,double &a,double &b,double &c){
31.     a = y.y - x.y;
32.     b = x.x - y.x;
33.     c = y.x * x.y - x.x * y.y;
34. }
35. inlinevoid initial(){
36.     p[1] = Point(0,0);
37.     p[2] = Point(0,1000000);
38.     p[3] = Point(1000000,1000000);
39.     p[4] = Point(1000000,0);
40.     p[5] = p[1];
41.     p[0] = p[4];
42.     cCnt = 4;
43. }
44. inline Point intersect(Point x,Point y,double a,double b,double c){
45. double u = fabs(a * x.x + b * x.y + c);
46. double v = fabs(a * y.x + b * y.y + c);
47. return Point( (x.x * v + y.x * u) / (u + v) , (x.y * v + y.y * u) / (u + v) );
48. }
49. inlinevoid cut(double a,double b ,double c){
50.     curCnt = 0;
51. if(a == 0 && b == 0 && c == 0){
52.         flag = 0;
53. return ;
54.     }
55. for(int i = 1; i <= cCnt; ++i){
56. if(sig(a*p[i].x + b*p[i].y + c) >= 0)q[++curCnt] = p[i];
57. //注意这里一定是大于0，而不是大于等于0，直接排除掉a = b = c = 0这种情况
58. else {
59. if(sig(a*p[i-1].x + b*p[i-1].y + c) > 0){
60.                 q[++curCnt] = intersect(p[i],p[i-1],a,b,c);
61.             }
62. if(sig(a*p[i+1].x + b*p[i+1].y + c) > 0){
63.                 q[++curCnt] = intersect(p[i],p[i+1],a,b,c);
64.             }
65.         }
66.     }
67. double area = 0;
68. for(int i = 1; i <= curCnt; ++i)
69.         area += p[i].x * p[i + 1].y - p[i + 1].x * p[i].y;
70.     area = fabs(area / 2.0);
71. if(area <= EPS)flag = 0;
```

```
72. for(int i = 1; i <= curCnt; ++i)p[i] = q[i];
73.     p[curCnt+1] = q[1];p[0] = p[curCnt];
74.     cCnt = curCnt;
75. }
76. Node nodes[MAXN];
77. int N;
78. int main(){
79.     scanf("%d",&N);
80. for(int i = 1; i <= N; ++i)
81.         nodes[i].input();
82. bool ok;
83. double a,b,c;
84. for(int i = 1; i <= N; ++i){
85.         ok = 1;
86.         initial();
87.         flag = 1;
88. for(int j = 1; j <= N; j++){
89. if(i == j)continue;
90.             getabc(a,b,c,nodes[i],nodes[j]);
91.             cut(a,b,c);
92. if(flag == 0){
93.                 ok = 0;
94. break;
95.             }
96.         }
97. if(ok)printf("Yes/n");
98. else printf("No/n");
99.     }
100. return 0;
101. }
```

POJ 2540 Hotter Colder

http://acm.pku.edu.cn/JudgeOnline/problem?id=2540

半平面交求线性规划可行区域的面积，需要求两点的中垂线。

注意直线切割多边形的时候，向新的点数组（q）里加点的时候，点的顺序必须和原来保持一致（顺时针）

[cpp]view plaincopy

```
1. #include
2. #include
3. #include
4. usingnamespace std;
5. #define EPS 1e-6
6. constint MAXN = 200;
7. struct Point{
8. double x,y;
9.     Point(){}
10.     Point(double _x,double _y):x(_x),y(_y){}
11. void input(){
12.         scanf("%lf%lf",&x,&y);
13.     }
14. };
15. Point p[MAXN],q[MAXN];
16. int cCnt ,curCnt;
17. inline Point intersect(Point x,Point y,double a,double b,double c){
18. double u = fabs(a * x.x + b * x.y + c);
19. double v = fabs(a * y.x + b * y.y + c);
20. return Point( (x.x * v + y.x * u) / (u + v) , (x.y * v + y.y * u) / (u + v) );
21. }
22. int sig(double k){
23. return (k < -EPS) ? -1 : (k > EPS);
24. }
25. inlinevoid gethalf(Point p1,Point p2,double &a,double &b,double &c,
26. int flag){
27. int fhalf;
```

```
    a = p2.x - p1.x;
    b = p2.y - p1.y;
    c = (p1.y*p1.y - p2.y*p2.y - p2.x*p2.x + p1.x*p1.x)/2;
double tmp = a*p2.x + b*p2.y + c;
if(flag == 1){
if(sig(tmp) >= 0)fhalf = 1;
else fhalf = -1;
    }
if(flag == -1){
if(sig(tmp) >= 0)fhalf = -1;
else fhalf = 1;
    }
if(fhalf == -1){
        a = -a;
        b = -b;
        c = -c;
    }
}
inlinevoid cut(double a,double b,double c){
    curCnt = 0;
for(int i = 1; i <= cCnt; ++i){
if(sig(a*p[i].x + b*p[i].y + c) >= 0)q[++curCnt] = p[i];
else {
if(sig(a*p[i-1].x + b*p[i-1].y + c) > 0){
                    q[++curCnt] = intersect(p[i-1],p[i],a,b,c);
                }
if(sig(a*p[i+1].x + b*p[i+1].y + c) > 0){
                    q[++curCnt] = intersect(p[i+1],p[i],a,b,c);
                }
            }
        }
for(int i = 1; i <= curCnt; ++i)
        p[i] = q[i];
    p[curCnt + 1] = p[1];
    p[0] = p[curCnt];
    cCnt = curCnt;
}
int main(){
    p[1] = Point(0,0);
    p[2] = Point(10,0);
    p[3] = Point(10,10);
    p[4] = Point(0,10);
    p[5] = p[1];
    p[0] = p[4];
    cCnt = 4;
double a1,b1;
char op[20];
bool mark = 1;
    Point last = Point(0,0);
while(scanf("%lf%lf%s",&a1,&b1,&op) != EOF){
        Point current = Point(a1,b1);
if(!mark){
            puts("0.00");
continue;
        }
if(!strcmp(op,"Same")){
            puts("0.00");
            mark = 0;
continue;
        }
int fhalf = 0,flag = 0;
if(!strcmp(op,"Hotter"))flag = 1;
elseif(!strcmp(op,"Colder"))flag = -1;
double a,b,c;
```

```
92.         gethalf(last,current,a,b,c,flag);
93.         cut(a,b,c);
94. double area = 0;
95. for(int i = 1; i <= cCnt; ++i)
96.             area += p[i].x * p[i + 1].y - p[i + 1].x * p[i].y;
97.         area = fabs(area / 2.0);
98.         printf("%.2lf/n",area);
99.         last = current;
100.     }
101. return 0;
102. }
```

另外还有一个数据比较多的，必须用nlogn的算法解决的半平面交的问题：

http://acm.pku.edu.cn/JudgeOnline/problem?id=2451

Zzy 专为他那篇nlogn算法解决半平面交问题的论文而出的题目，至今不会自己写。