2025/11/29 19:01 1/3 0.声明

## 0.声明

#### 1.点的存储

点的保存为直角坐标系,如下

```
class Point{
public:
    ld x , y; //ld = long double
};
```

#### 2.边的存储

边的保存为两个点\$(p,v)\$\_其中\$p\$为直线上某一点\_\$v\$为方向向量\_\$angle\$是直线的倾角(用于排序)

```
class Line{
public:
    Point p , v;
    ld angle;
};
```

### 3.函数使用

求两直线交点(调用前需保证两直线不能平行)

```
Point Linewithline(Line a , Line b){
   Point u = a.p - b.p;
   ld t = (b.v * u) / (a.v * b.v);
   return a.p + a.v * t;
}
```

判断一个点是否在某一条直线对应的半平面内(严格内部)

```
bool Bothside(Point p , Line a){
   return a.v * (p - a.p) > eps;
}
```

# 1.引入

在一个无限大的平面上,有\$n\$条直线。每条直线相当于一次切割,只保留目前平面中这条直线左边或者右边的部分,求最终剩余平面。

由于每条直线的切割可以看成一个半平面,最终平面可以看成这\$n\$个半平面的交。

# 2.算法介绍

### 1.朴素的做法

update: 2020/2021:teams:running\_chicken:halfplaneintersection https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:running\_chicken:halfplaneintersection&rev=1589545790

### 此做法不做为讲述重点。

我们可以存储半平面对应的所有交点和交线,对于每一个新的半平面,我们暴力的枚举每一条线,并把这条线和半平面对应的分界线求交点。

当合法的交点数恰好为2时,我们可以根据方向留下那一半的直线和点,删去另一半直线和点,加入这条 直线和两个新的交点。

否则对应这个半平面我们不做任何处理。

该方法时间复杂度为\$O(n^2)\$

此方法由于复杂度较高,在大部分题目中都无法使用,因此在这里不提供代码了。

### 2.优化的做法

在朴素的做法中,我们对于最终的半平面交是一个凸包的这个性质并没有使用。如果我们有效的利用这个性质,将能得到时间复杂度更低的做法。

首先我们按照半平面的分界线的方向角度\$angle\$排序,对于方向角相同的半平面,我们留那个半平面面积更小的直线(即所留的直线在另外一条直线的内部)。

我们尝试去维护一个下凸壳(这个下凸壳包围的区域就是目前得到的半平面)。

对于一个新加的半平面(对应直线\$L\_i\$□\我们尝试将目前下凸壳的点\$P\_{tail}\$(下凸壳的点是相邻两组成直线的交点)和直线\$L\_i\$进行比较□\$P\_{tail}\$如果不在\$L\_i\$一侧,就说明该交点和下凸壳最后一条直线已经没有任何的意义(这条线严格在最终交平面外面),将这条线删除,并继续和前一交点\$P\_{tail-1}\$比较,直到得到下凸壳中无点或者点严格在\$L\_i\$一侧。同时在下凸壳靠前的点,我们也通过\$L\_i\$往后筛,如果点\$P\_{head}\$不符合要求(和前面一样),删除并继续比较,直到下凸壳中无点或者点严格在\$L i\$一侧。

在过程中要同时维护下凸壳的点集合\$P\$和边集合\$Q\$□

最后再用下凸壳中的第一条边,从后往前删除不符合要求的点。

最终\$P\$和\$Q\$即为所求

该方法时间复杂度为\$O(nlogn)\$

代码

```
void HalfPlaneIntersect(Line *L){
    int head = 0 , tail = 0;
    Q[head] = L[1];
    for(int i = 2 ; i <= n ; ++ i){
        if(fabs(L[i].angle - L[i - 1].angle) < eps) continue; //两条直线平行
        while(head < tail && !Bothside(P[tail - 1] , L[i]))
            tail --;
    while(head < tail && !Bothside(P[head] , L[i]))
            head ++;
    Q[++ tail] = L[i];
    if(head < tail)
        P[tail - 1] = Linewithline(Q[tail] , Q[tail - 1]);
    }
}</pre>
```

https://wiki.cvbbacm.com/ Printed on 2025/11/29 19:01

2025/11/29 19:01 3/3 0.声明

```
while(head < tail && !Bothside(P[tail - 1] , Q[head]))
    tail --;
P[tail] = Linewithline(Q[tail] , Q[head]);
}</pre>
```

From:

https://wiki.cvbbacm.com/ - CVBB ACM Team

Permanent link:

 $https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:running\_chicken:halfplane intersection \& rev=1589545790.$ 



