习题16

* 1. 凸包问题

1.1.1.算法描述

首先确定一点，横坐标最大的点是边界点，之后依次进行找下两个点（求其他点都在一边），进行收纳。

1.1.2.伪代码

convex\_hull

//输入：点集

//输出：凸包连接点

//求出最大边点

While(点没有全被收纳）

求解直线方程

判断其余点是否在一边

如果在一边，收纳

1.1.3.算法实现

struct point{

int x,y;

bool haspack=0;

};

vector<point> ans;

bool validate(point\* p,int len,point p\_tmp,point p\_max){

//求直线方程y=k\*x+b

int k=(p\_tmp.y-p\_max.y)/(p\_tmp.x-p\_max.x);

int b=p\_tmp.y-k\*p\_tmp.x;

int sign=0;

for(int i=0;i<len;i++){

if(!p[i].haspack&&!(p[i].x==p\_tmp.x&&p[i].y==p\_tmp.y)){

//求是否在一边

if(!sign){

sign=k\*p[i].x+b-p[i].y;

}else{

if(sign\*(k\*p[i].x+b-p[i].y)<0) return false;

}

}

}

return true;

}

void convex\_hull(point\* p,int len){

point& p\_max=p[0];

for(int i=0;i<len;i++)

if(p\_max.x<p[i].x) p\_max=p[i];//找出横坐标最边的点

p\_max.haspack=1;//表为收纳

ans.push\_back(p\_max);

bool change=true;

while(change){

change=false;

for(int i=0;i<len;i++){

if(!p[i].haspack){//未被收纳

bool vali=validate(p,len,p[i],p\_max);//校验其他点

if(vali){

p[i].haspack=true;

ans.push\_back(p[i]);

change=true;

break;

}

}

}

}

}

1.1.4.测试用例

int main()

{

point p[5];

p[0].x=0;p[0].y=0;

p[1].x=1;p[1].y=1;

p[2].x=1;p[2].y=-1;

p[3].x=2;p[3].y=1;

p[4].x=2;p[4].y=0;

convex\_hull(p,5);

system("pause");

return 0;

}

2.1.实验小结

这次凸包问题，我们使用数学的方法来求解凸包，很多时候，数学方式来解决问题也是一种很好的办法