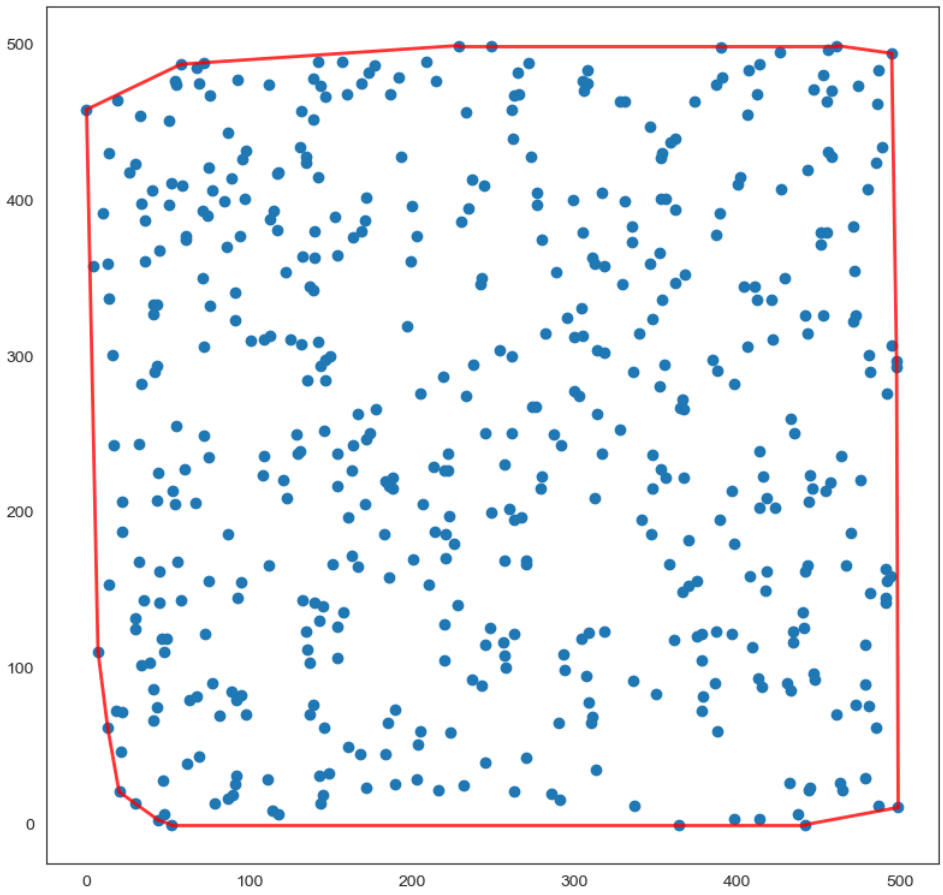
算法设计与分析实验

计算机85 张子诚

实验三

一.问题描述

给定一个二维平面点集S,试从中找出最小的点集C,使其形成的Convex Hull包含点集S中的所有点。



注：上图为算法可视化后的结果

Convex Hull of a point set P

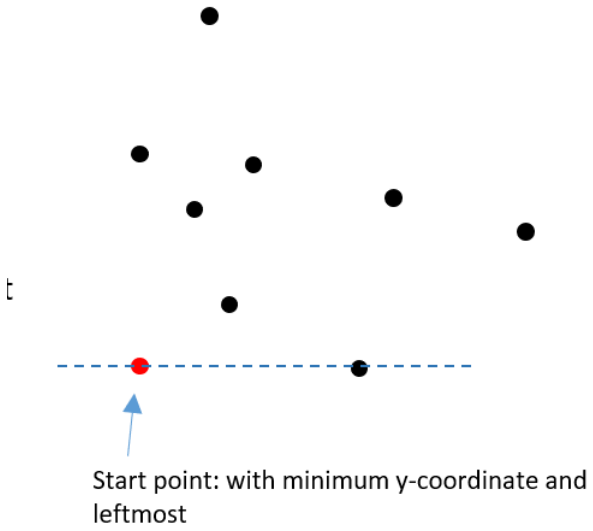
- Smallest convex set containing P
- Intersection of all convex sets containing P

二.算法设计与分析

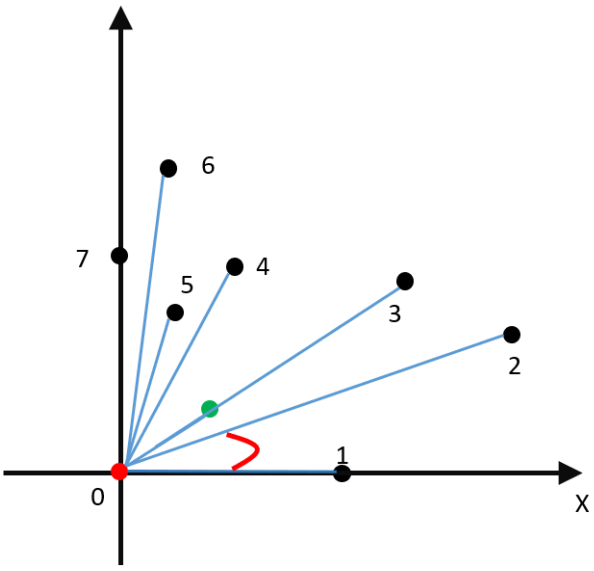
算法描述

本文在解决该问题的时候参考了Graham scan算法，该算法的基本思想是利用了贪心思想，该算法首先选择最坐下角的点P_0，然后以P_0 为极点，水平向右为极轴，对其余点按极角排序，然后依次遍历所有排序的点，检查他们是否符合逆时针旋向。

step1

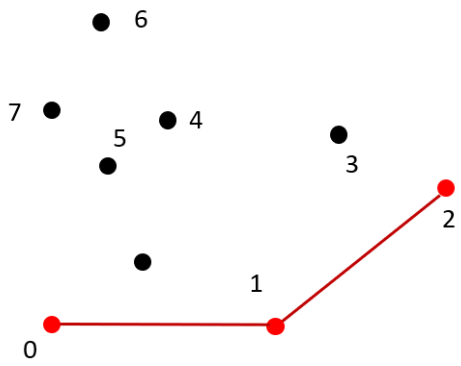


step2

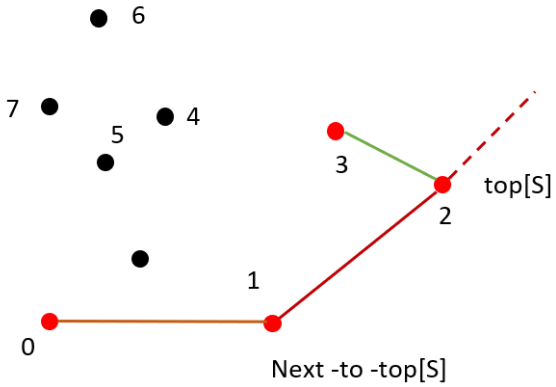


step3

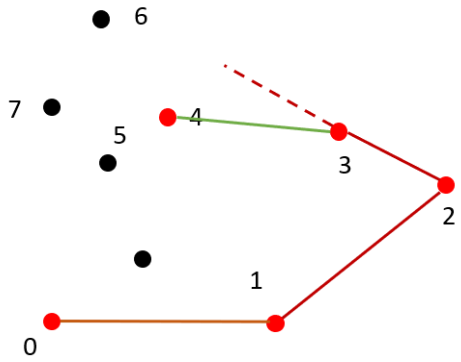
- 将最初三个点加入栈中



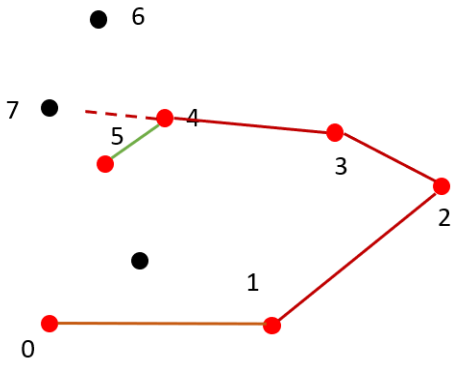
Stack S: <p0, p1, p2>



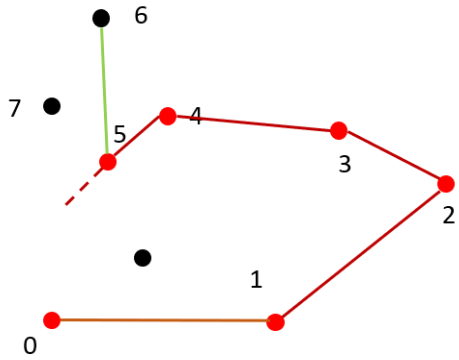
Stack S: <p0, p1, p2>

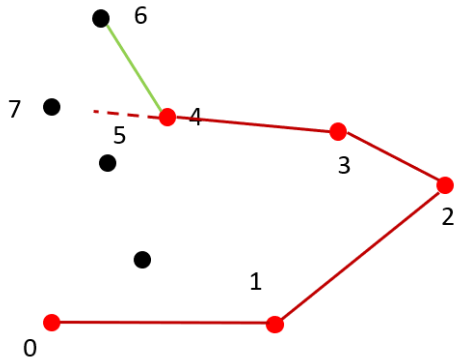


Stack S: <p0, p1, p2, p3>

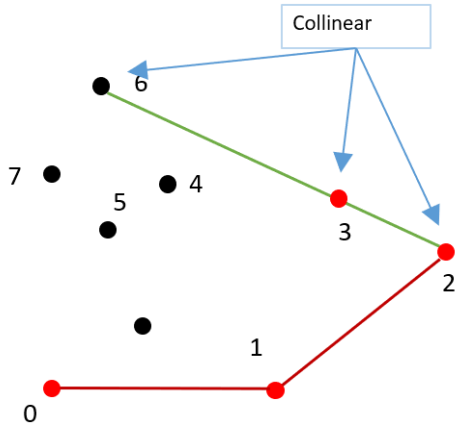


Stack S: <p0, p1, p2, p3, p4>

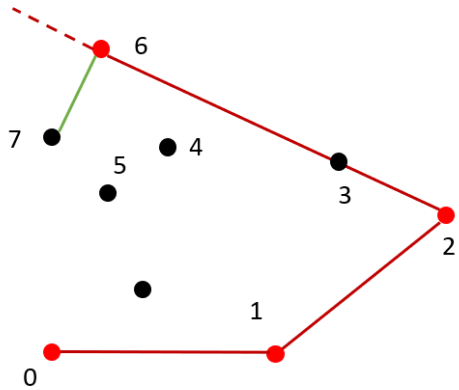




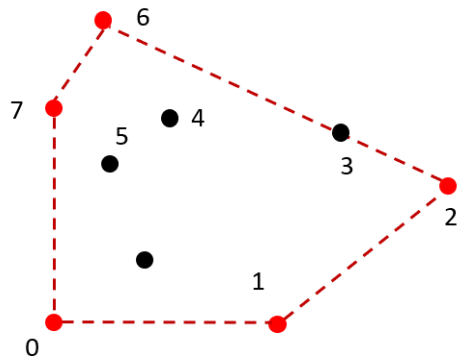
Stack S: <p0, p1, p2, p3, p4>



Stack S: <p0, p1, p2, p3>



Stack S: <p0, p1, p2>



Persudo-code

```
let points be the list of points
let stack = empty_stack()

find the lowest y-coordinate and leftmost point, called P0
sort points by polar angle with P0, if several points have the same polar angle then only keep the farthest

for point in points:
    # pop the last point from the stack if we turn clockwise to reach this point
    while count stack > 1 and ccw(next_to_top(stack), top(stack), point) <= 0:
        pop stack
    push point to stack
end
```

算法正确性

> 参考 https://www.comp.nus.edu.sg/~rahul/allfiles/cs6234-16-convexhull.pptx

我们的任务是证明下面两个引理

1. 栈中的点总是构成凸多边形的顶点
2. 从栈中弹出的点一定不是凸包的顶点，并且被包含在一个新的凸包中

引理一证明

首先base condition 有三个顶点构成三角形

栈的变化有两种情况

- case1： 从中弹出顶点
- case2： 向栈中压入顶点

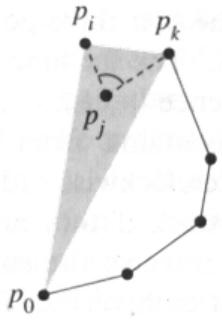
case1：

我们根据一个几何性质可以证明该操作保持凸性：如果从一个凸多边行中删去一个顶点，那么余下的多边形依然是凸的

case2：如果p_i 压入栈中，意味着我们的旋向依然是一致的所以$p_0, p_1, \cdots, p_{i-1}, p_i$ 仍构成凸多边形

从而引理一得到证明

引理二证明



假设p_j 从栈中弹出，因为如上图所示p_j 改变了旋转方向，因为我们的顶点都是按照

极角排序的，所以存在三角形$p_0p_ip_k$ 使得 p_j 要么在三角形内部，要么在p_i,p_k的连线上，任意一种情况都表明p_j 不是凸包上的顶点

从而引理二得到证明

结论:

因为每个从栈中弹出的顶点都不是凸包中的顶点，而栈中的顶点总构成凸多边形，所以算法正确

源代码

```
import random
from plotConvexHull import *


def read_file(path,mode):
    with open(path,mode=mode,encoding='utf-8') as file:
        lines=file.readlines()
        table=dict()
        key=0
        for item in lines:
            temp=list(map(int,item.strip('\n').split(' ')))
            if len(temp)==1:
                key=temp[0]
                table[key]=[]
            else:
                table[key].append(temp)

        for key,val in table.items():
            table[key]=np.array(val)
            # 检查是否读入错误
            assert len(val)==key

    return table

def output_file(path,mode):
    with open(path,mode=mode,encoding='utf-8') as file:
        lines=file.readlines()
        for item in lines:
            temp=item.strip('\n')
            print(temp)







class Sol(object):
    @staticmethod
    def ConvexHull(points):
```

```
        # 比较方法
        def method(p):
            vec1=np.array([1,0])
            vec2=np.array([p[0]-start[0],p[1]-start[1]])
            return vec1.dot(vec2)/(np.linalg.norm(vec2))
        # 通过外积来判断是否是逆时针方向
        def ccw(x,y,z):
            vec1,vec2=y-x,z-y
            return np.cross(vec1,vec2)

        # 寻找起始点 O(n)
        index=np.argmin(points[:,1])
        start=points[index,:]
        points=np.delete(points,index,axis=0) # 删除起始节点
        # 对点集按照角度排序
        points=np.array(sorted(points,key=method,reverse=True))
        points=np.concatenate((start.reshape(1,2),points))

        # 创建栈 初始化
        stack=[]
        # 做cross product 来判断 方向
        for i in range(len(points)):
            while len(stack)>1 and ccw(stack[-2],stack[-1],points[i,:]) <0:
                stack.pop()
            stack.append(points[i,:])

        return np.array(stack)

    @staticmethod
    def ConvexHull_dynamic(points):
        # 排序的比较方法
        def method(p):
            vec1=np.array([1,0])
            vec2=np.array([p[0]-start[0],p[1]-start[1]])
            # 返回的是cos值
            return vec1.dot(vec2)/(np.linalg.norm(vec2))
        # 通过外积来判断是否是逆时针方向
        def ccw(x,y,z):
            vec1,vec2=y-x,z-y
            return np.cross(vec1,vec2)

        # 寻找起始点 O(n)
        index=np.argmin(points[:,1])
        start=points[index,:]
        points=np.delete(points,index,axis=0) # 删除起始节点
        # 对点集按照角度排序
        points=np.array(sorted(points,key=method,reverse=True)) # cos 值单调递减
        points=np.concatenate((start.reshape(1,2),points))
        # 交互图像
        plt.ion()
        # 创建栈 初始化
        stack=[]
        # 做cross product 来判断 方向
        for i in range(len(points)):
            while len(stack)>1 and ccw(stack[-2],stack[-1],points[i,:]) <=0:
                stack.pop()
                plot_process(points,np.array(stack))
            stack.append(points[i,:])
            plot_process(points,np.array(stack))

        plot_result(points,np.array(stack))
```

```
        plt.pause(10)



if __name__=='__main__':
    # 随机生成点集
    # num=50
    # points=np.random.uniform(low=0.0,high=10.0,size=(num,2))
    # ans=np.array(Sol.ConvexHull(points))
    # print(ans)
    # # 绘制原始图像
    # plot_result(points,ans)
    table=read_file('./exp3(2)_in.txt','r')
    for key,val in table.items():
        Sol.ConvexHull(val)
    output_file('./exp3(2)_out.txt','r')

```

结果输出

```
PS D:\study\algorithm\ex\ex3> & C:/ProgramData/Anaconda3/python.exe d:/study/algorithm/ex/ex3/ex3.py
[6, 8, 2, 9, 0, 5, 3]
[10, 8, 0, 14, 19, 7, 23, 21, 17, 2]
[25, 9, 15, 37, 11, 18, 36, 41, 22]
[28, 22, 36, 86, 65, 33, 62, 81, 49, 53, 4, 45]
[15, 164, 114, 118, 156, 72, 16, 19, 105, 21, 78, 4, 129, 176]
[274, 288, 247, 179, 58, 22, 29, 10, 160, 36, 212, 48, 196]
[149, 94, 34, 27, 370, 191, 308, 247, 320, 273, 52, 352, 39, 354]
[440, 347, 5, 44, 124, 91, 287, 10, 251, 166, 196, 282, 296, 271, 416, 2]
[475, 152, 310, 313, 189, 529, 374, 466, 399, 689, 109]
```

结果与输出文件一致