# 計算機演算法作業 2
# Convex Hull

班級:資訊三甲

學號:D0617735

姓名:霍湛軒

# 規則建立說明

一開始先把點根據x及y進行排序，這樣可以從下而上左到右開始對點進行掃描，而在掃描的過程中不會出現共線的情況。

利用叉積找出線之間的內角度，判斷是否為凸線。

順序的掃描只能找到下半部的凸多邊形，要找到上半部的話要反序掃描。最後得到多邊形頂點的集合。

# 程式碼

```
import random
import operator


class Point(object):

    def __init__(self, x, y):
        self.x = x
        self.y = y

    @staticmethod
    def cross(o, a, b):#find inner angle from this three points
        return (a.x - o.x) * (b.y - o.y) - (a.y - o.y) * (b.x -
o.x)

    @staticmethod
    def findConvexHull(pointsList):
        #sort points by x and y
        pointsList = sorted(pointsList,
key=operator.attrgetter('x', 'y'))
        numOfVextexs = 0
        vextexs = []

        #find lower convex hull
        for i in range(len(pointsList)):
            #have nore than two vextexs and check next point's
inner angle is getter than 180
            while (numOfVextexs >= 2 and
Point.cross(vextexs[numOfVextexs - 2], vextexs[numOfVextexs - 1],
pointsList[i]) <= 0):
                #if not pop the last vertex
                vextexs.pop()
                numOfVextexs -= 1
            vextexs.append(pointsList[i])
            numOfVextexs += 1

        #find upper convex hull
```

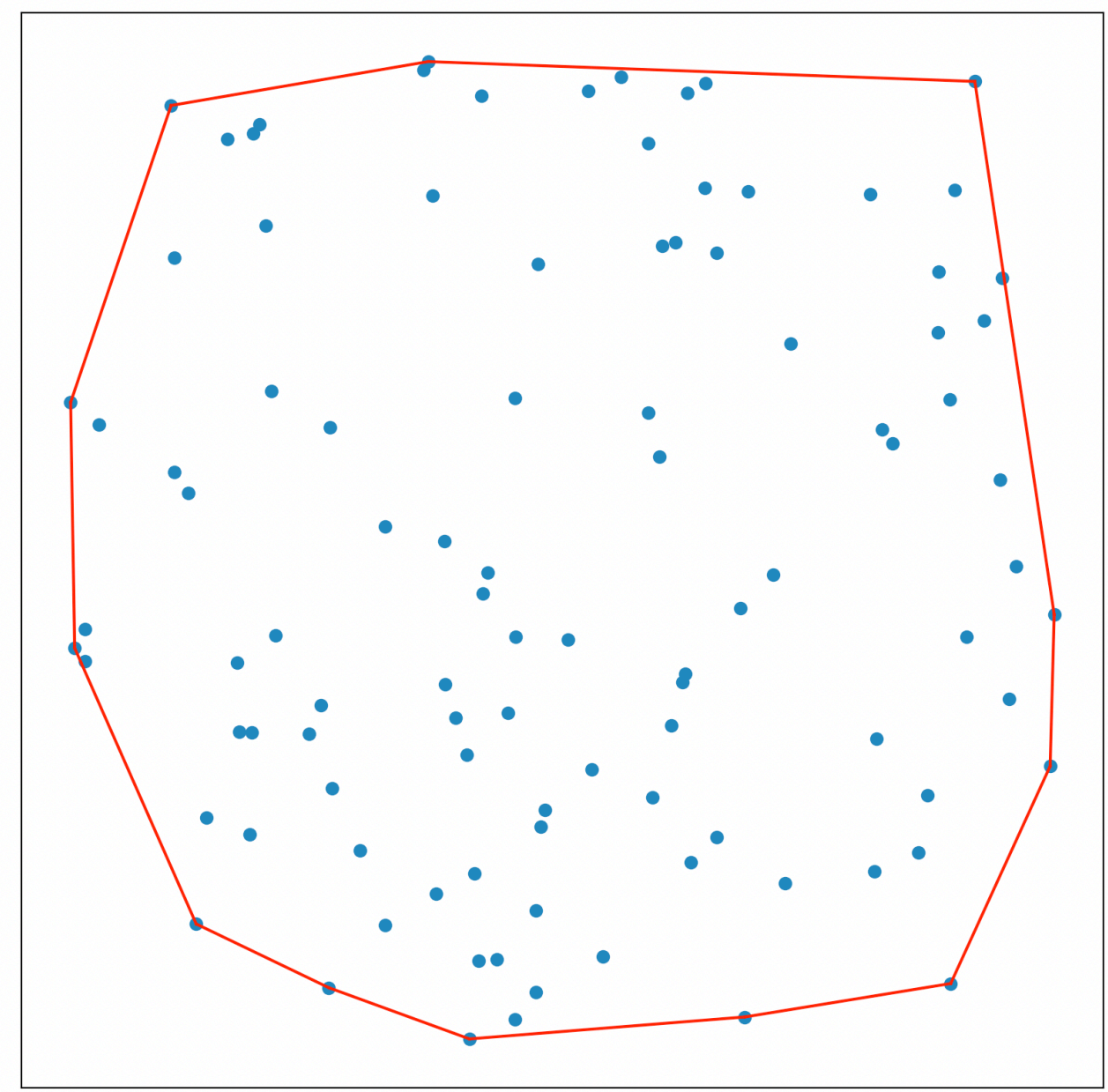
```
        start = numOfVextexs + 1
        for i in range(len(pointsList) - 2, -1, -1):
            while (numOfVextexs >= start and
Point.cross(vextexs[numOfVextexs - 2], vextexs[numOfVextexs - 1],
pointsList[i]) <= 0):
                vextexs.pop()
                numOfVextexs -= 1
            vextexs.append(pointsList[i])
            numOfVextexs += 1

        return vextexs
```

```
    @staticmethod
    def randomPoints(size):
        pointList = []
        index = 0
        while(size != index):
            x = random.randint(0, 1000)
            y = random.randint(0, 1000)
            p = Point(x, y)
            if (p in pointList):
                continue
            pointList.append(p)
            index += 1
        return pointList
```

```
    def __str__(self):
        return "(" + str(self.x) + "," + str(self.y) + ")"
```

```
    def __repr__(self):
        return str(self)
```

# 執行結果



# 心得：

這個演算比我想像中的要簡單一點，只要跟著邏輯就可以把程式碼慢慢推演出來。今我對演算法改觀。