Experiment 9

AA Experiment DATE:

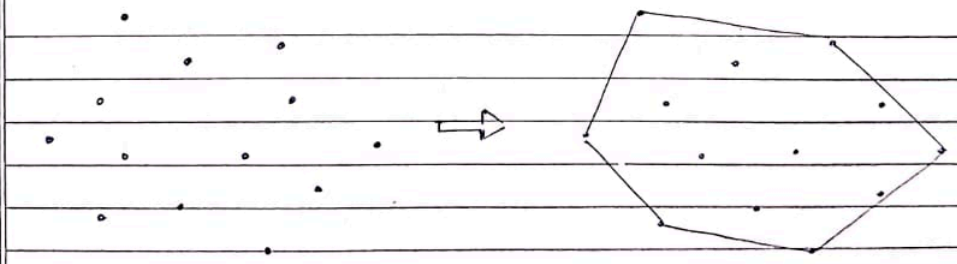
Kreena Shah
60004210243
C'32

Aim : To understand & implement Graham's scan algorithm (Computational Geometry)

Theory :
The Graham Scan algorithm is a simple & efficient algo for computing the convex hull of a set of points.
It works by iteratively adding points to the convex hull until all points have been added.

Implementation
(1) Sort points
(2) Accept / Reject Points

A convex polygon is a polygon in which all interior angles are less than 180°
A convex hull can be constructed for any set of point, regardless of arrangement



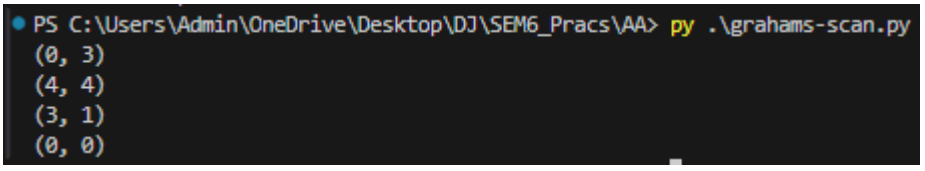
Code :

```
from functools import cmp_to_key

class Point:
        def __init__(self, x = None, y = None):
                self.x = x
```

```
                self.y = y

p0 = Point(0, 0)

def nextToTop(S):
        return S[-2]

def distSq(p1, p2):
        return ((p1.x - p2.x) * (p1.x - p2.x) +
                        (p1.y - p2.y) * (p1.y - p2.y))

def orientation(p, q, r):
        val = ((q.y - p.y) * (r.x - q.x) -
                (q.x - p.x) * (r.y - q.y))
        if val == 0:
                return 0
        elif val > 0:
                return 1
        else:
                return 2

def compare(p1, p2):
        o = orientation(p0, p1, p2)
        if o == 0:
                if distSq(p0, p2) >= distSq(p0, p1):
                        return -1
                else:
                        return 1
        else:
                if o == 2:
                        return -1
                else:
                        return 1

def convexHull(points, n):
        ymin = points[0].y
        min = 0
        for i in range(1, n):
                y = points[i].y
                if ((y < ymin) or
                        (ymin == y and points[i].x < points[min].x)):
                        ymin = points[i].y
                        min = i

        points[0], points[min] = points[min], points[0]

        p0 = points[0]
        points = sorted(points, key=cmp_to_key(compare))
```

```
        m = 1
        for i in range(1, n):
                while ((i < n - 1) and
                (orientation(p0, points[i], points[i + 1]) == 0)):
                        i += 1
                points[m] = points[i]
                m += 1

        if m < 3:
                return

        S = []
        S.append(points[0])
        S.append(points[1])
        S.append(points[2])

        for i in range(3, m):
                while ((len(S) > 1) and
                (orientation(nextToTop(S), S[-1], points[i]) != 2)):
                        S.pop()
                S.append(points[i])

        while S:
                p = S[-1]
                print("(" + str(p.x) + ", " + str(p.y) + ")")
                S.pop()

input_points = [(0, 3), (1, 1), (2, 2), (4, 4),
                                (0, 0), (1, 2), (3, 1), (3, 3)]
points = []
for point in input_points:
        points.append(Point(point[0], point[1]))
n = len(points)
convexHull(points, n)
```

Output :



DATE:

Conclusion : During Graham Scan implementation for convex hull computation, challenges arose, notably handling collinear points efficiently.
To address this, I integrated a sorting mechanism based on polar angles from a reference point, facilating easy identificati & removal of collinear points.
The other challenge was handling degenerate cases such as duplicate points & points with identical polar angles relative to reference point.
Furthermore, optimizing the algorithms performance to handle large datasets efficiently while maintaining correctness was another significant challenge
Iterative testing led to the identification of most effective approach, ensuring accurate convex hull generation.