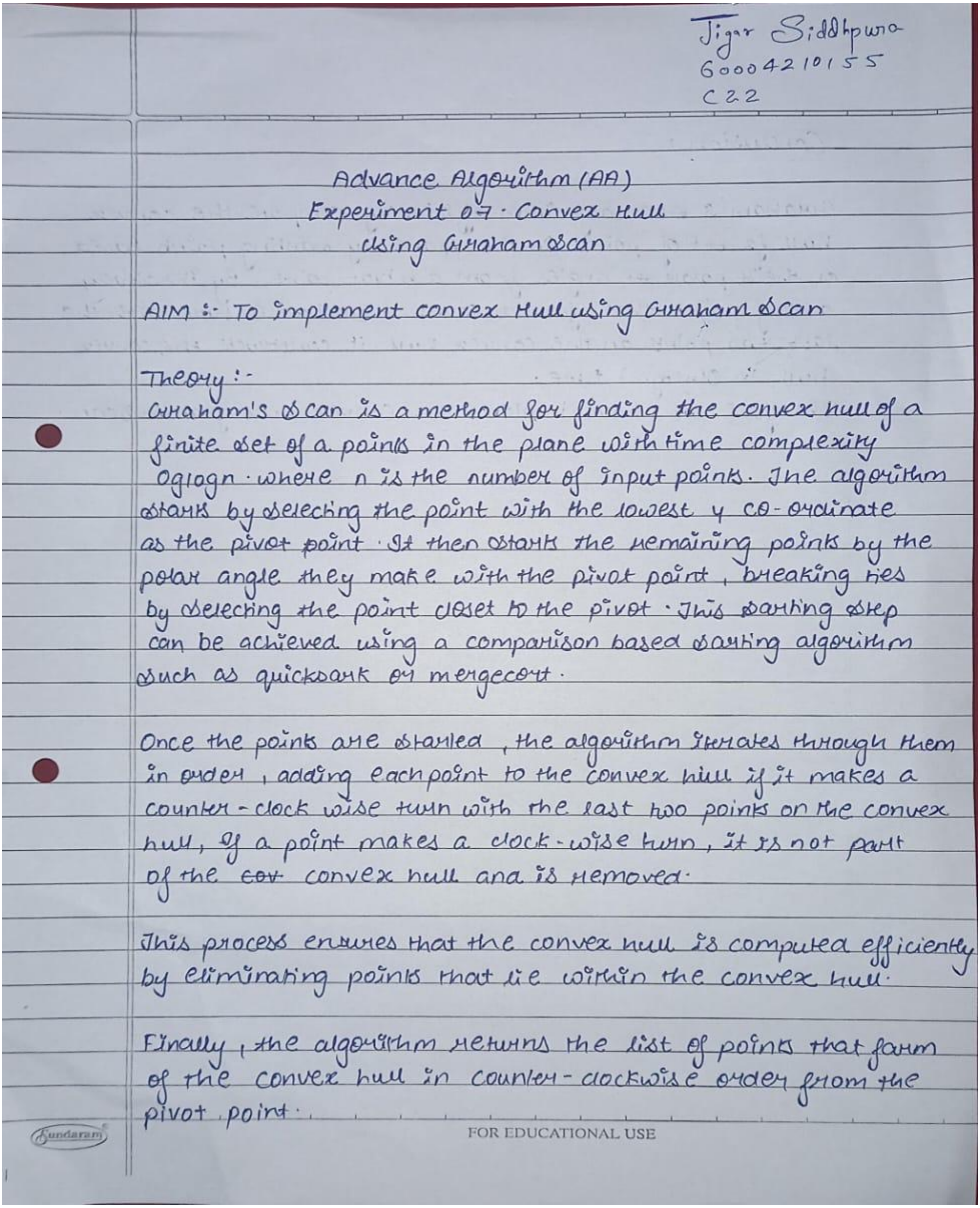
Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

AA - Experiment 7 - Convex Hull using Graham Scan



Jigar Siddhpura
60004210155
C22

Advance Algorithm (AA)
Experiment 07 · Convex Hull
using Graham Scan

AIM :- To implement convex Hull using Graham Scan

Theory :-
Graham's Scan is a method for finding the convex hull of a finite set of a points in the plane with time complexity O(nlogn) · where n is the number of input points. The algorithm starts by selecting the point with the lowest y co-ordinate as the pivot point. It then starts the remaining points by the polar angle they make with the pivot point, breaking ties by selecting the point closet to the pivot. This sorting step can be achieved using a comparison based sorting algorithm such as quicksort or mergecort.

Once the points are started, the algorithm iterates through them in order, adding each point to the convex hull if it makes a counter-clock wise turn with the last two points on the convex hull, if a point makes a clock-wise turn, it is not part of the ~~cov~~ convex hull and is removed.

This process ensures that the convex hull is computed efficiently by eliminating points that lie within the convex hull.

Finally, the algorithm returns the list of points that form of the convex hull in counter-clockwise order from the pivot point.

Conclusion :-

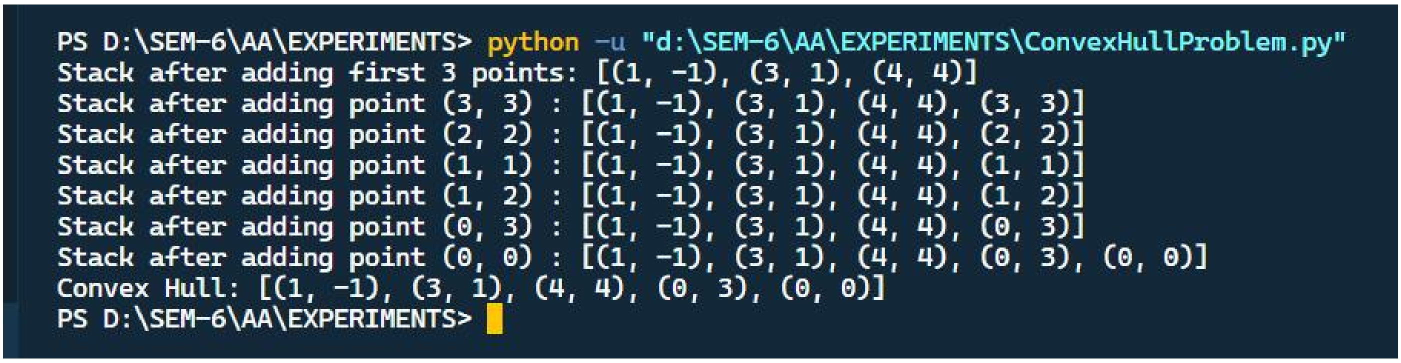
Graham's scan algorithm efficiently compares the convex hull fa set of points in the plane by sarting points based on their polar ~~et~~ angles from a pivot point. By iteratively adding points that forma counter-clockwise turn with the last two points on the convex hull, it constructs the convex hull in O(nlogn) time.

Therefore we have implemented convex hull graham scan.

CODE :

```
import math

def orientation(p, q, r):
    val = (q[1] - p[1]) * (r[0] - q[0]) - (q[0] - p[0]) * (r[1] - q[1])
    if val == 0:
        return 0  # Collinear
    return 1 if val > 0 else 2  # Clockwise or counterclockwise

def graham_scan(points):
    n = len(points)
    if n < 3:
        return []
    min_point = min(points, key=lambda x: (x[1], x[0]))
    sorted_points = sorted(points, key=lambda x: (math.atan2(x[1] - min_point[1], x[0] - min_point[0]), x))
    convex_hull_stack = [sorted_points[0], sorted_points[1], sorted_points[2]]
    print("Stack after adding first 3 points:", convex_hull_stack)

    for i in range(3, n):
        while len(convex_hull_stack) > 1 and orientation(convex_hull_stack[-2], convex_hull_stack[-1],
sorted_points[i]) != 2:
            convex_hull_stack.pop()
        convex_hull_stack.append(sorted_points[i])
        print("Stack after adding point", sorted_points[i], ":", convex_hull_stack)
    return convex_hull_stack

points = [(0, 3), (2, 2), (1, 1),(4,4), (1, 2),(3,1), (0, 0), (3, 3),(1,-1)]
convex_hull = graham_scan(points)
print("Convex Hull:", convex_hull)
```

OUTPUT :



```
PS D:\SEM-6\AA\EXPERIMENTS> python -u "d:\SEM-6\AA\EXPERIMENTS\ConvexHullProblem.py"
Stack after adding first 3 points: [(1, -1), (3, 1), (4, 4)]
Stack after adding point (3, 3) : [(1, -1), (3, 1), (4, 4), (3, 3)]
Stack after adding point (2, 2) : [(1, -1), (3, 1), (4, 4), (2, 2)]
Stack after adding point (1, 1) : [(1, -1), (3, 1), (4, 4), (1, 1)]
Stack after adding point (1, 2) : [(1, -1), (3, 1), (4, 4), (1, 2)]
Stack after adding point (0, 3) : [(1, -1), (3, 1), (4, 4), (0, 3)]
Stack after adding point (0, 0) : [(1, -1), (3, 1), (4, 4), (0, 3), (0, 0)]
Convex Hull: [(1, -1), (3, 1), (4, 4), (0, 3), (0, 0)]
PS D:\SEM-6\AA\EXPERIMENTS>
```