MOTH ERADICATION

Batch-13

B.V.L.Kavya : 21B01A6111 : AIML
V.Valli Suvarna : 21B01A04B4 : ECE-B
K.Roopa : 21B01A0330 : MECHANICAL
K.Swetha : 21B01A5453 : AIDS-A
A.Kundanika: 21B01A0304 : MECHANICAL

February 4, 2023

Introduction

The task is to determine the minimum length of the perimeter of the polygon which encloses all the traps within it. This listing's starting point is not important, but it must be clockwise and start and end at the same location.

Approach

- We started solving the problem by importing sys and math modules to implement the distance and orientation functions.
- The ***distance*** function calculates the Euclidean distance between points given as input through the command line arguments in a 2D plane.
- The ***orientation*** function is used to determine the orientation of three points p, q, and r in a 2D plane by calculating the cross product of vectors formed by the points.
- The ***convexhull*** function computes the convex hull of a set of points in a 2D plane using Gift Wrapping algorithm and the main function will be executed, processing the command-line arguments passed to the script.

Learnings

- We discovered that working together as a team is highly beneficial so that we can share all of our ideas for a better execution of our code.
- We learned how to employ user-defined functions like distance, convexhull and orientation.
- We learned about the fundamental computational geometry problem known as the ***convexhull*** problem and used the $\boldsymbol{gift - wrapping}$ approach to solve it.
- We discovered how to use methods like append and format.
- We learned about the git commands.
- We gained knowledge about how to use LaTeX to create presentations.

Challenges

- We have faced the difficulty while dealing with the ***Convexhull*** problem
- We encountered some difficulties while utilising the ***GiftWrapping*** algorithm to solve the convex hull problem.
- We encountered some issues when examining the clockwise condition that is necessary for the given code.

Statistics

- The Python code has overall 65 lines.
- Two modules are imported.
  They are:
  1. sys module
  2. math module
- The code has 4 user-defined functions.
  They are:
  1. distance()
  2. orientation()
  3. convexhull()
  4. main()
- The code has the following methods.
  They are:
  1. append()
  2. format()

Demo/Screenshots

The Convex Hull problem is fundamental problem in computational geometry and deals with finding the smallest convex polygon that encloses a set of points in a plane. The solution to this problem, the Convex Hull, has applications in a variety of fields including computer graphics, pattern recognition, and robotics. The Convex Hull can be found using algorithms such as Jarvis March, Graham's scan, and the Quickhull method.

Figure 1: Convex Hull

Demo/Screenshots

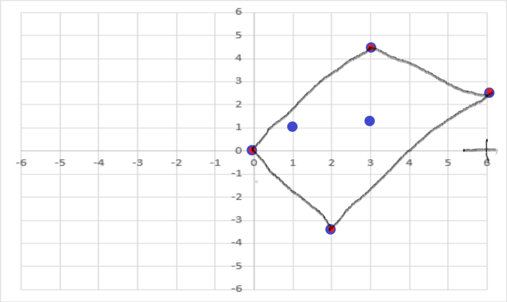

Figure 2: Representation of Convex Hull

Demo/Screenshots

C: > Users > HP > Documents > writings > motheradication.py

```python
import sys
import math

def distance(p1, p2):
    return math.sqrt((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2)

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
    elif val > 0:
        return 1
    else:
        return 2

```

Directory of the code

Modules imported

Distance function which calculates the distance between two points.

The orientation function determines the orientation of the points given in the plane

Figure 3: Code part-1

Demo/Screenshots

```python
def convex_hull(points):
    n = len(points)
    if n < 3:
        return []
    hull = []
    l = 0
    for i in range(1, n):
        if points[i][0] < points[l][0]:
            l = i
    p = l
    q = 0
    while True:
        hull.append(points[p])
        q = (p + 1) % n
        for i in range(n):
            if orientation(points[p], points[i], points[q]) == 2:
                q = i
        p = q
        if p == l:
            break
    return hull

```

The convex hull is defined as the smallest convex polygon that contains all the points in the set. The algorithm used to find the convex hull is the Jarvis March.

Figure 4: Code part-2

Demo/Screenshots

```python
def main(argv):
    region = 0
    n = int(argv[0])
    if n == 0:
        return
    region += 1
    points = []
    for i in range(n):

        x, y = map(float, argv[i * 2 + 1: i * 2 + 3])
        points.append((x, y))
    hull = convex_hull(points)
    print("Region #{}:".format(region))
    for i, p in enumerate(hull):
        print("({:.1f}, {:.1f})".format(p[0], p[1]), end="")
        if i == len(hull) - 1:
            print("-({:.1f}, {:.1f})".format(hull[0][0], hull[0][1]))
        else:
            print("-", end="")
    perimeter = 0
    for i in range(len(hull) - 1):
        perimeter += distance(hull[i], hull[i + 1])
    perimeter += distance(hull[-1], hull[0])
    print("Perimeter length = {:.2f}".format(perimeter))
    print()

if __name__ == "__main__":
    main(sys.argv[1:])
```

The main function is the entry point of the program where it takes the input through the command line arguments.

Figure 5: Code part-3

Demo/Screenshots

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP\Documents\writings> python motheradication.py 3 1 2 4 10 5 12.3
Region #1:
(1.0, 2.0)-(5.0, 12.3)-(4.0, 10.0)-(1.0, 2.0)
Perimeter length = 22.10

PS C:\Users\HP\Documents\writings> python motheradication.py 6 0 0 1 1 3.1 1.3 3 4.5 6 2.1 2 -3.2
Region #2:
(0.0, 0.0)-(2.0, -3.2)-(6.0, 2.1)-(3.0, 4.5)-(0.0, 0.0)
Perimeter length = 19.66

PS C:\Users\HP\Documents\writings> python motheradication.py 7 1 0.5 5 0 4 1.5 3 -0.2 2.5 -1.5 0 0 2 2
Region #3:
(0.0, 0.0)-(2.5, -1.5)-(5.0, 0.0)-(4.0, 1.5)-(2.0, 2.0)-(0.0, 0.0)
Perimeter length = 12.52
```

Figure 6: Final Output

THANK YOU