Hamiltonian Path Application
Travelling Salesman Problem

Project By:
Anmol Gupta - 15D070025
Kavya Prudhvi - 15D070014
Mohak Sahu - 15D070047
Harshit Khariwal - 15D070026

Problem Statement

A salesman wants to visit a number of cities which are connected. The distance between any two cities is known. Which is the order in which the salesman should visit the cities so that he minimizes the total distance we walks and simultaneously the fuel consumption of the salesman’s car gets minimized? The problem is to find Hamiltonian path with minimum weight.

An equivalent formulation in terms of graph theory is: Given a complete weighted graph (where the vertices would represent the cities, the edges would represent the roads, and the weights would be the cost or distance of that road), find a Hamiltonian path with the least weight.

Overall Algorithm

Insight into algorithm for finding a Hamiltonian Path:
A Hamiltonian Path in a graph having N vertices is nothing but a permutation of the vertices of the graph [v1, v2, v3, ...vN-1, vN] , such that there is an edge between vi and vi+1 where 1 ≤ i ≤ N-1. So it can be checked for all permutations of the vertices whether any of them represents a Hamiltonian Path or not.

Constraints:
1.We have introduced weighted directed edges as the distance between the cities.
2.We have assigned weights to each city i.e. halt time at each city.
3.We have implemented the same in PyGame and the result of which is shown in the video in the zip file.

Algorithm for discovering Hamiltonian paths

Used Depth First Traversal to find all possible hamiltonian paths between all possible pairs of cities connected with a road (an edge), stored all of them in a list. But since we are only interested in a path which allows the postman to cover every city, only those paths are chosen which have ‘n’ (the no. of cities) elements.
Now, considering the time taken (equivalently the distance) by the postman to travel through all the roads(edges), the path which demands the least amount of time is chosen as the most optimized path the postman should take to deliver his post in least amount of time.

Basic outline of Code

The program takes a graph as an input in the form a dictionary whose keys are the vertices and the values are the corresponding neighboring vertices.

It identifies the edges in the graph, and subsequently asks for the respective weights of each.

On inputting the edges, it finds all the possible Hamiltonian paths in the code, and then chooses the smallest one of them.

Steps to run the code

1. Run the code in command prompt by writing: python “AdvancedTSP.py”.
   It asks the user to enter the location of the city to be visited in graph format as ‘vertices : [all its neighbouring vertices] form’.

2. Enter the Sample input of dictionary. An example of the same is:
   { "a" : ["d","e"],"b" : ["c","d"],"c" : ["b", "d", "e"],"d" : ["a","b", "c"],"e" : ["c","a"]}

3. It identifies and display the edges in the graph, and subsequently asks for the respective lengths of each path. Care has to be taken to enter the weights of edges (i,j) and (j,i) as same, otherwise it will return a faulty output.
   A sample input of lengths (corresponding to the sample input of vertices provided above) is:
   [1,2,3,4,5,3,6,5,2,1,6,4]

4. This displays the assigned lengths to each path.

Continued....

5. Then It asks to enter the average speed of the salesman.
   Sample input - ‘22’

6. Then It asks to enter the average mileage of the salesman’s car in km/litre..
   Sample input - ‘30’

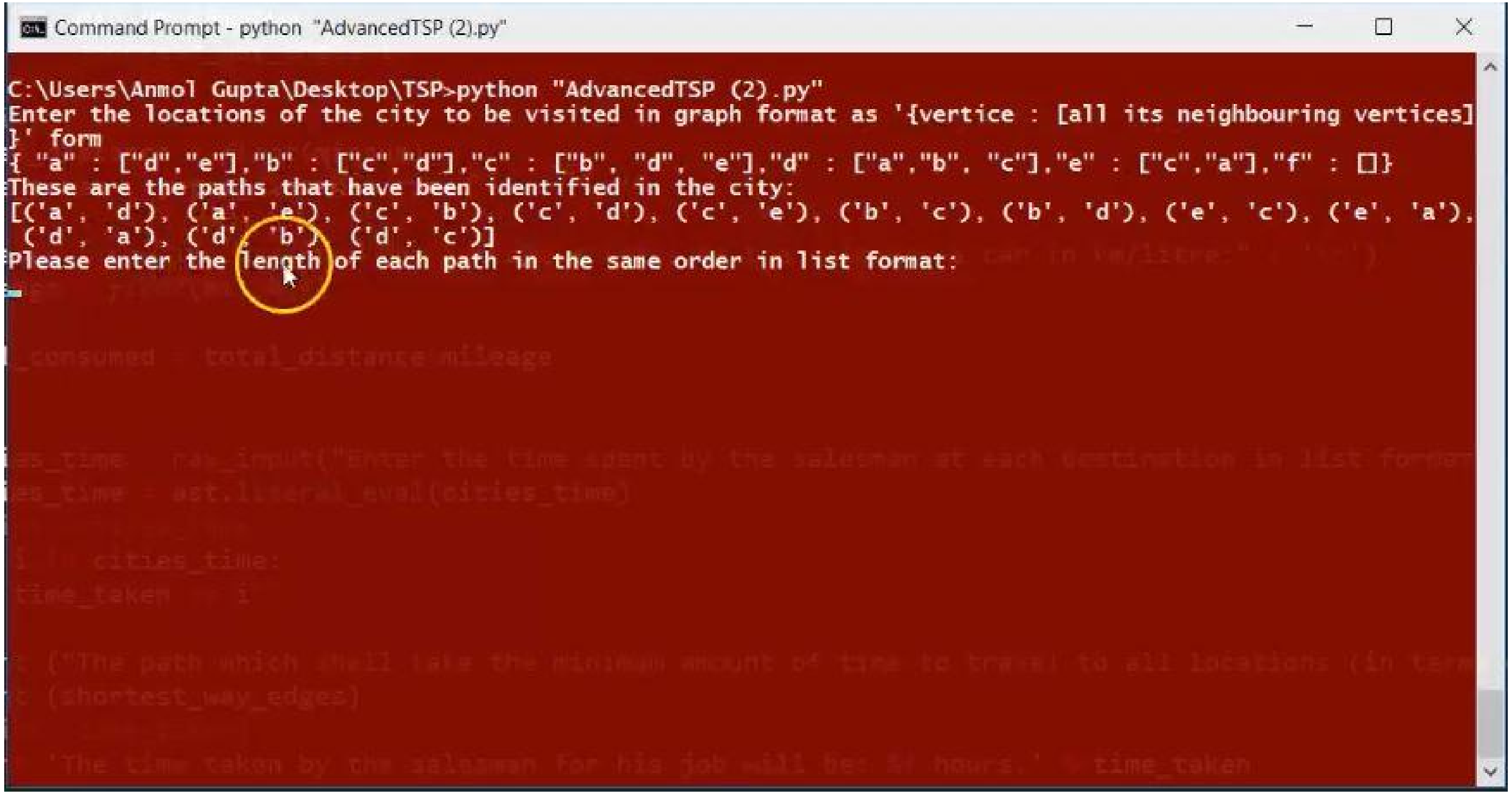
7. Then enter the time spent by the salesman at each destination in list format.
   Sample Input - [0.2,0.3,0.1,0.5,0.2]

8. Now It displays the path which shall take minimum amount of time to travel to all locations(in terms of path), time taken by the salesman to complete this path and the fuel consumption during the journey. And It asks whether we want to start PyGame or not?
Type ‘Y’ for Yes and ‘N’ for No.
Typing Y will display the PyGame representation of the above problem statement.

RESULTS

The code was tested using 5 examples, the inputs for which have been given in the uploaded folder (the coded inputs as well as the graph images)

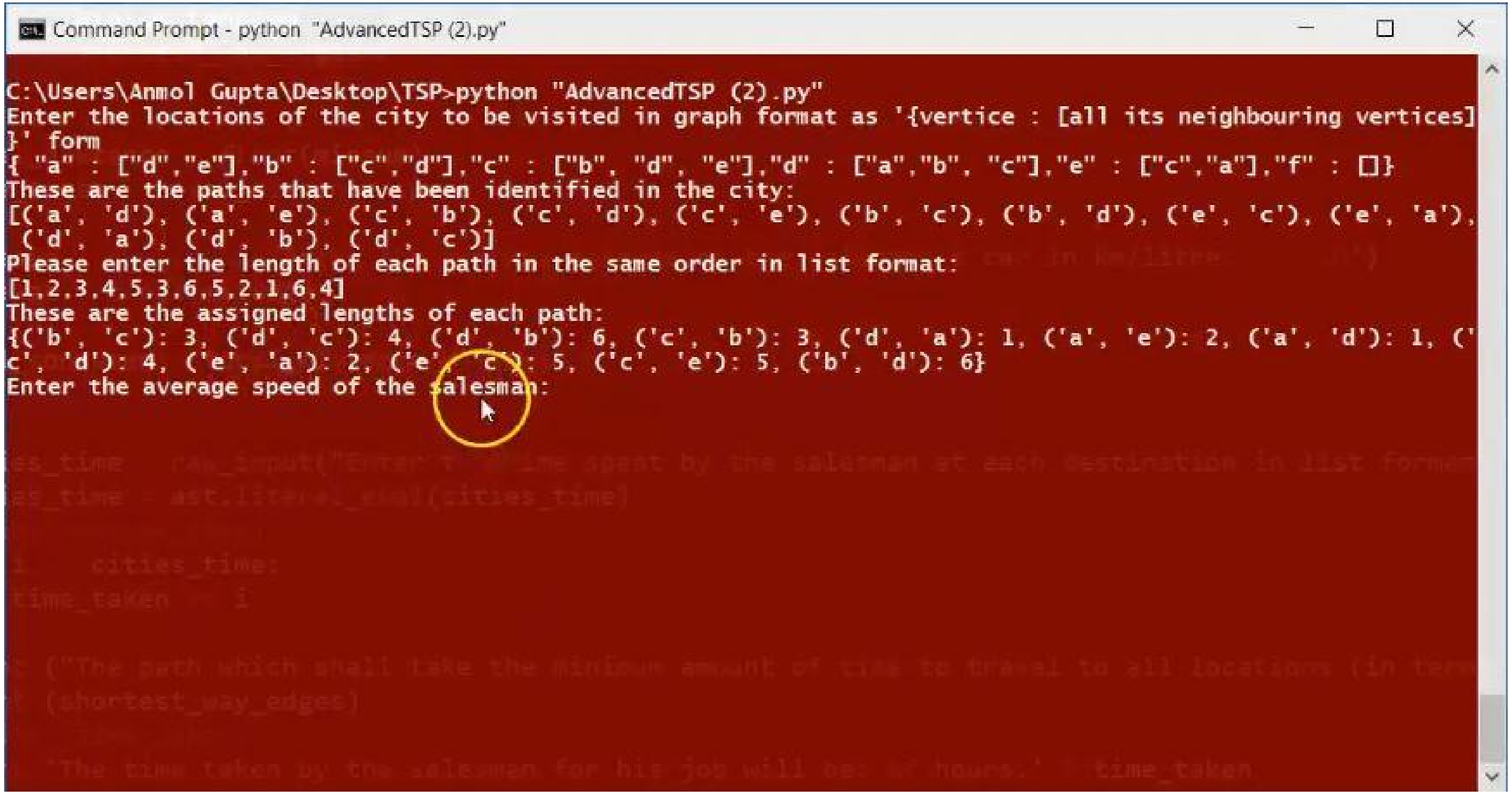
1. Identification of paths from the graph



Command Prompt - python "AdvancedTSP (2).py"

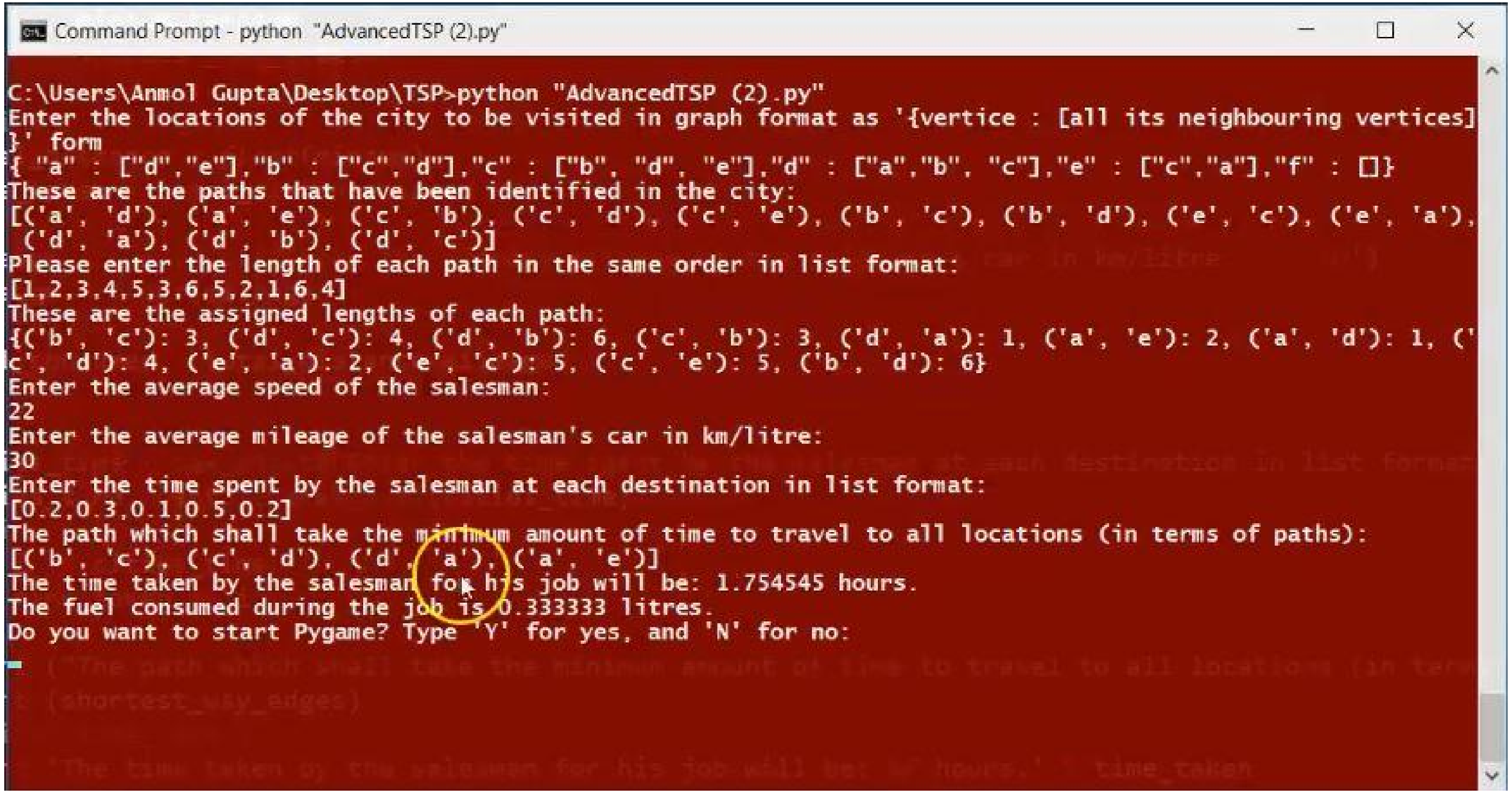
```
C:\Users\Anmol Gupta\Desktop\TSP>python "AdvancedTSP (2).py"
Enter the locations of the city to be visited in graph format as '{vertice : [all its neighbouring vertices]
}' form
{ "a" : ["d","e"],"b" : ["c","d"],"c" : ["b", "d", "e"],"d" : ["a","b", "c"],"e" : ["c","a"],"f" : []}
These are the paths that have been identified in the city:
[('a', 'd'), ('a', 'e'), ('c', 'b'), ('c', 'd'), ('c', 'e'), ('b', 'c'), ('b', 'd'), ('e', 'c'), ('e', 'a'),
 ('d', 'a'), ('d', 'b'), ('d', 'c')]
Please enter the length of each path in the same order in list format:
```

2. Assignment of lengths to paths



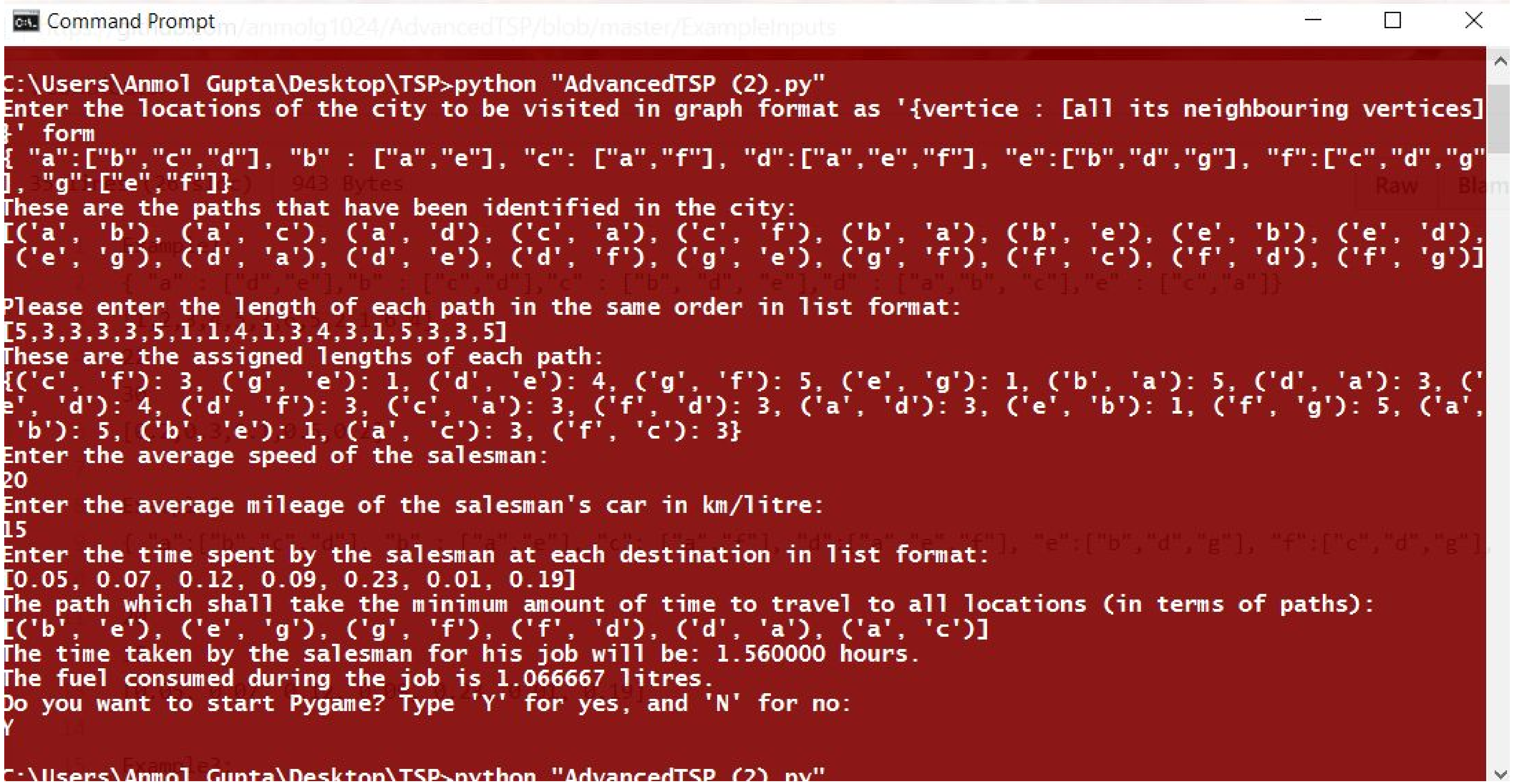
```
C:\Users\Anmol Gupta\Desktop\TSP>python "AdvancedTSP (2).py"
Enter the locations of the city to be visited in graph format as '{vertice : [all its neighbouring vertices]
}' form
{ "a" : ["d","e"],"b" : ["c","d"],"c" : ["b", "d", "e"],"d" : ["a","b", "c"],"e" : ["c","a"],"f" : []}
These are the paths that have been identified in the city:
[('a', 'd'), ('a', 'e'), ('c', 'b'), ('c', 'd'), ('c', 'e'), ('b', 'c'), ('b', 'd'), ('e', 'c'), ('e', 'a'),
 ('d', 'a'), ('d', 'b'), ('d', 'c')]
Please enter the length of each path in the same order in list format:
[1,2,3,4,5,3,6,5,2,1,6,4]
These are the assigned lengths of each path:
{('b', 'c'): 3, ('d', 'c'): 4, ('d', 'b'): 6, ('c', 'b'): 3, ('d', 'a'): 1, ('a', 'e'): 2, ('a', 'd'): 1, ('
c', 'd'): 4, ('e', 'a'): 2, ('e', 'c'): 5, ('c', 'e'): 5, ('b', 'd'): 6}
Enter the average speed of the salesman:
```

3. Path with minimum time



```
C:\Users\Anmol Gupta\Desktop\TSP>python "AdvancedTSP (2).py"
Enter the locations of the city to be visited in graph format as '{vertice : [all its neighbouring vertices]
}' form
{ "a" : ["d","e"],"b" : ["c","d"],"c" : ["b", "d", "e"],"d" : ["a","b", "c"],"e" : ["c","a"],"f" : []}
These are the paths that have been identified in the city:
[('a', 'd'), ('a', 'e'), ('c', 'b'), ('c', 'd'), ('c', 'e'), ('b', 'c'), ('b', 'd'), ('e', 'c'), ('e', 'a'),
 ('d', 'a'), ('d', 'b'), ('d', 'c')]
Please enter the length of each path in the same order in list format:
[1,2,3,4,5,3,6,5,2,1,6,4]
These are the assigned lengths of each path:
{('b', 'c'): 3, ('d', 'c'): 4, ('d', 'b'): 6, ('c', 'b'): 3, ('d', 'a'): 1, ('a', 'e'): 2, ('a', 'd'): 1, ('
c', 'd'): 4, ('e', 'a'): 2, ('e', 'c'): 5, ('c', 'e'): 5, ('b', 'd'): 6}
Enter the average speed of the salesman:
22
Enter the average mileage of the salesman's car in km/litre:
30
Enter the time spent by the salesman at each destination in list format:
[0.2,0.3,0.1,0.5,0.2]
The path which shall take the minimum amount of time to travel to all locations (in terms of paths):
[('b', 'c'), ('c', 'd'), ('d', 'a'), ('a', 'e')]
The time taken by the salesman for his job will be: 1.754545 hours.
The fuel consumed during the job is 0.333333 litres.
Do you want to start Pygame? Type 'Y' for yes, and 'N' for no:
```

3.2 Example2:

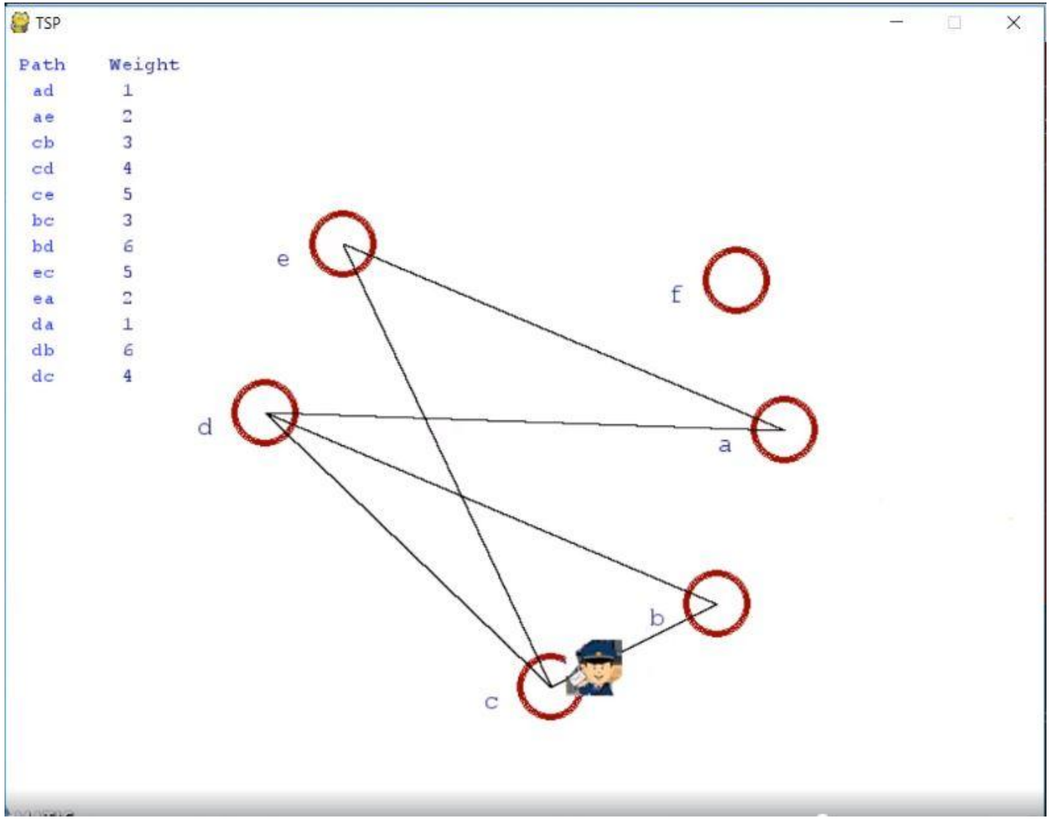


```
C:\Users\Anmol Gupta\Desktop\TSP>python "AdvancedTSP (2).py"
Enter the locations of the city to be visited in graph format as '{vertice : [all its neighbouring vertices]
}' form
{ "a":["b","c","d"], "b" : ["a","e"], "c": ["a","f"], "d":["a","e","f"], "e":["b","d","g"], "f":["c","d","g"
], "g":["e","f"]}
These are the paths that have been identified in the city:
[('a', 'b'), ('a', 'c'), ('a', 'd'), ('c', 'a'), ('c', 'f'), ('b', 'a'), ('b', 'e'), ('e', 'b'), ('e', 'd'),
 ('e', 'g'), ('d', 'a'), ('d', 'e'), ('d', 'f'), ('g', 'e'), ('g', 'f'), ('f', 'c'), ('f', 'd'), ('f', 'g')]

Please enter the length of each path in the same order in list format:
[5,3,3,3,3,5,1,1,4,1,3,4,3,1,5,3,3,5]
These are the assigned lengths of each path:
{('c', 'f'): 3, ('g', 'e'): 1, ('d', 'e'): 4, ('g', 'f'): 5, ('e', 'g'): 1, ('b', 'a'): 5, ('d', 'a'): 3, ('
e', 'd'): 4, ('d', 'f'): 3, ('c', 'a'): 3, ('f', 'd'): 3, ('a', 'd'): 3, ('e', 'b'): 1, ('f', 'g'): 5, ('a',
 'b'): 5, ('b', 'e'): 1, ('a', 'c'): 3, ('f', 'c'): 3}
Enter the average speed of the salesman:
20
Enter the average mileage of the salesman's car in km/litre:
15
Enter the time spent by the salesman at each destination in list format:
[0.05, 0.07, 0.12, 0.09, 0.23, 0.01, 0.19]
The path which shall take the minimum amount of time to travel to all locations (in terms of paths):
[('b', 'e'), ('e', 'g'), ('g', 'f'), ('f', 'd'), ('d', 'a'), ('a', 'c')]
The time taken by the salesman for his job will be: 1.560000 hours.
The fuel consumed during the job is 1.066667 litres.
Do you want to start Pygame? Type 'Y' for yes, and 'N' for no:
Y

C:\Users\Anmol Gupta\Desktop\TSP>python "AdvancedTSP (2).py"
```

4. PyGame Representation



Path	Weight
ad	1
ae	2
cb	3
cd	4
ce	5
bc	3
bd	6
ec	5
ea	2
da	1
db	6
dc	4

References

- http://www.techeuler.com/python/usage-of-underscores-before-and-after-function-name-in-python/
- https://www.python-course.eu/graphs_python.php
- https://stackoverflow.com/questions/17264174/python-how-to-take-a-dictionary-as-input
- https://pythonprogramming.net/
- http://www.nerdparadise.com/programming/pygame/part1
- https://www.hackerearth.com/practice/algorithms/graphs/hamiltonian-path/tutorial/
- http://www.csd.uoc.gr/~hy583/reviewed_notes/euler.pdf

Work Distribution

Anmol Gupta (15D070025)	Ideation, Basic Graph structuring, Additional features, Documentation
Mohak Sahu (15D070047)	Ideation, data structures, Implementation of algorithm
Kavya Prudhvi (15D070014)	Ideation, pygame, output animation and representation
Harshit Khariwal (15D070026)	Ideation, Data structures, Documentation

Thank you