Artificial Intelligence
(CS13217)

Lab Report

Name:	Iqra Arshad
Registration #:	CSU-S15-120
Lab Report #:	05
Dated:	18-05-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 5
Greedy Algorithms

Objective
To understand and implement the Greeedy Algorithm Problem.

Software Tool
1. Operating System Window 10
2. Sublime Version 3.0
3. Python

1 Theory

Dijkstra's algorithm, conceived by computer scientist Edsger Dijkstra and published in 1959, Dijkstra's algorithm is called the single-source shortest path. It is also known as the single source shortest path problem. It computes length of the shortest path from the source to each of the remaining vertices in the graph.
The single source shortest path problem can be described as follows:
1. Create a set sptSet that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.
2. Assign a distance value to all vertices . Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
3. While sptSet doesnt include all vertices.

```
Path: ('A', 'B', 2)
Path: ('A', 'G', 6)
Path: ('B', 'C', 7)
Path: ('B', 'E', 2)
Path: ('B', 'A', 2)
Path: ('C', 'F', 3)
Path: ('C', 'D', 3)
Path: ('E', 'F', 2)
Path: ('E', 'G', 1)
Path: ('E', 'B', 2)
Path: ('F', 'E', 2)
Path: ('F', 'H', 2)
Path: ('G', 'H', 4)
Path: ('H', 'D', 2)
Path: ('H', 'F', 2)
Path: ('D', 'C', 3)
Path: ('D', 'H', 2)
A -> D:
(10, ('D', ('H', ('F', ('E', ('B', ('A', ())))))))
[Finished in 0.4s]
```

Figure 1: Time Independent Feature Set

2 Task

2.1 Procedure: Task 1

Dijkstras algorithm can be used to determine the shortest path from one node in a graph to every other node within the same graph data structure, this Algorithm will run until all vertices in the graph have been visited.This means that the shortest path between any 2 nodes can be saved and looked up later.

2.2 Procedure: Task 2

```
from collections import defaultdict
from heapq import *

def dijkstra(edges, f, t):
    g = defaultdict(list)
    for l,r,c in edges:
        g[l].append((c,r))


    q, seen = [(0,f,())], set()
```

```
    while q:
        (cost,v1,path) = heappop(q)
        if v1 not in seen:
            seen.add(v1)
            path = (v1, path)

            if v1 == t: return (cost, path)

            for c, v2 in g.get(v1, ()):
                if v2 not in seen:
                    heappush(q, (cost+c, v2, path))

    return float("inf")
if __name__ == "__main__":
    edges = [
        ("A", "B", 2),
        ("A", "G", 6),
        ("B", "C", 7),
        ("B", "E", 2),
        ("B", "A", 2),
        ("C", "F", 3),
        ("C", "D", 3),
        ("E", "F", 2),
        ("E", "G", 1),
        ("E", "B", 2),
        ("F", "E", 2),
        ("F", "H", 2),
        ("G", "H", 4),
        ("H", "D", 2),
        ("H", "F", 2),
        ("D", "C", 3),
        ("D", "H", 2)
    ]
    for x in range (len(edges)):
        print "Path:",edges[x]
    print "A -> D:"
    print dijkstra(edges, "A", "D")
```

3 Conclusion

Hence we have successfully implemented a graph to solve the greedy Algorithm problem.