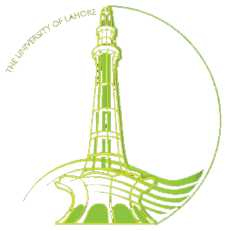

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Artificial Intelligence
(CS13217)

Lab Report

Name:	Iqra Arshad
Registration #:	CSU-S15-120
Lab Report #:	06
Dated:	18-05-2018
Submitted To:	Mr. Usman Ahmed

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

Experiment # 6
Greedy Algorithms (Prims Minimum Spanning Tree (MST)

Objective
To understand and implement the Greedy Algorithm.

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

1 Theory

In computer science, Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. ... However, running Prim's algorithm separately for each connected component of the graph, it can also be used to find the minimum spanning forest..
We start from one vertex and keep adding edges with the lowest weight until we we reach our goal.
The steps for implementing Prim's algorithm are as follows:
1. Initialize the minimum spanning tree with a vertex chosen at random.
2. Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree.
3. Keep repeating step 2 until we get a minimum spanning tree.

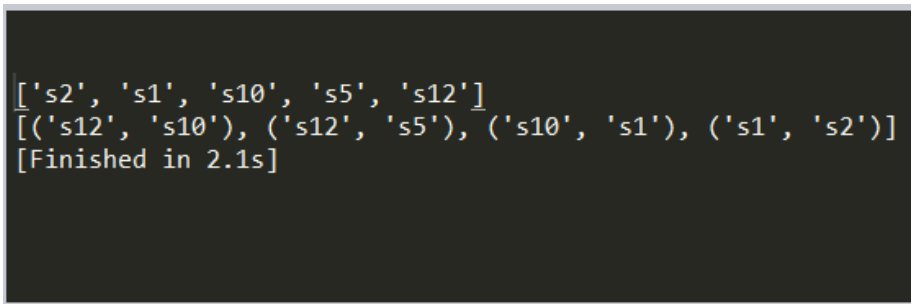

Figure 1: Time Independent Feature Set

2 Task

2.1 Procedure: Task 1

Prim's algorithm can be used to find the Minimum spanning tree in a graph ,a spanning tree means all vertices must be connected. So the two disjoint subsets (discussed above) of vertices must be connected to make a Spanning Tree. And they must be connected with the minimum weight edge to make it a Minimum Spanning Tree.

2.2 Procedure: Task 2

```
def prim(graph, root):
    assert type(graph)==dict

    nodes = graph.keys()
    print nodes
    nodes.remove(root)

    visited = [root]
    path = []
    next = None

    while nodes:
        distance = float('inf')
```

```
        for s in visited:
            for d in graph[s]:
                if d in visited or s == d:
                    continue
                if graph[s][d] < distance:
                    distance = graph[s][d]
                    pre = s
                    next = d
        path.append((pre, next))
        visited.append(next)
        nodes.remove(next)

    return path


if __name__ == '__main__':
    graph_dict = {  "s1":{"s1": 0, "s2": 2, "s10": 3, "s12": 4, "s5":3},
                    "s2":{"s1": 1, "s2": 0, "s10": 4, "s12": 2, "s5":2},
                    "s10":{"s1": 2, "s2": 6, "s10": 0, "s12":3, "s5":4},
                    "s12":{"s1": 3, "s2": 5, "s10": 2, "s12":0,"s5":2},
                    "s5":{"s1": 3, "s2": 5, "s10": 2, "s12":4,"s5":0},
    }

    path = prim(graph_dict, 's12')
    print path
```

3 Conclusion

We have thus successfully find a Minimum spanning tree using Prim's Algorithm.