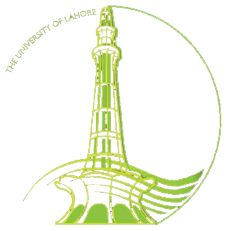

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Artificial Intelligence
(CS13217)

Lab Report

Name:	Wardha kanwal
Registration #:	CSU-S15-103
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
Implementation of Prim's Algorithm

Objective
To understand and implement the Prim's Algorithm problem.

Software Tool
1. Python
2. Sublime, version 3.0
3. Operating System, window 8.1

1 Theory

Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which
form a tree that includes every vertex
has the minimum sum of weights among all the trees that can be formed from the graph
It falls under a class of algorithms called greedy algorithms which find the local optimum in the hopes of finding a global optimum.
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:
Initialize the minimum spanning tree with a vertex chosen at random.
Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree
Keep repeating step 2 until we get a minimum spanning tree

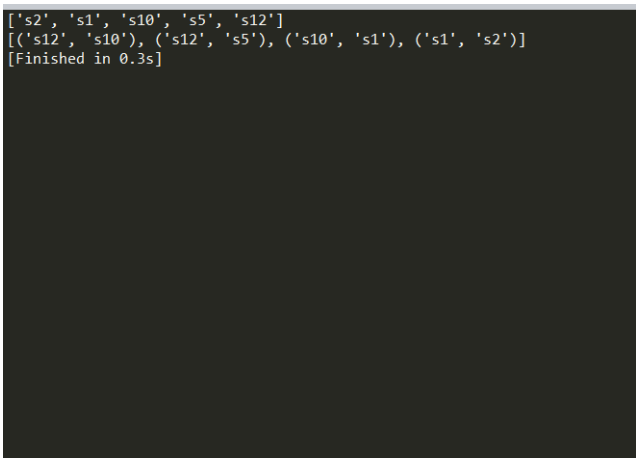

Figure 1: Time Independent Feature Set

2 Task

2.1 Procedure: Task 1

Select any vertex
Select the shortest edge connected to that vertex
Select the shortest edge connected to any vertex already connected
Repeat step 3 until all vertices have been connected

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

Prims algorithm is a vertex based algorithm
Prims algorithm Needs priority queue for locating the nearest vertex. The choice of priority queue matters in Prim implementation.
o Array optimal for dense graphs
o Binary heap better for sparse graphs
o Fibonacci heap best in theory, but not in practice.