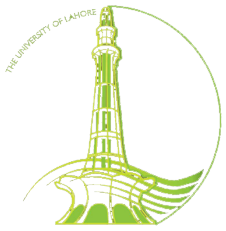

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Artificial Intelligence
(CS13217)

Lab Report

Name:	Wardha kanwal
Registration #:	CSU-S15-103
Lab Report #:	07
Dated:	25-05-2018
Submitted To:	Mr. Usman Ahmed

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

Experiment # 7
Implementing Kruskals Algorithm

Objective
To understand and implement the Kruskals problem.

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

1 Theory

Kruskal'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 the edges with the lowest weight and keep adding edges until we we reach our goal.

The steps for implementing Kruskal's algorithm are as follows:
-Sort all the edges from low weight to high
-Take the edge with the lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject this edge.
-Keep adding edges until we reach all vertices.

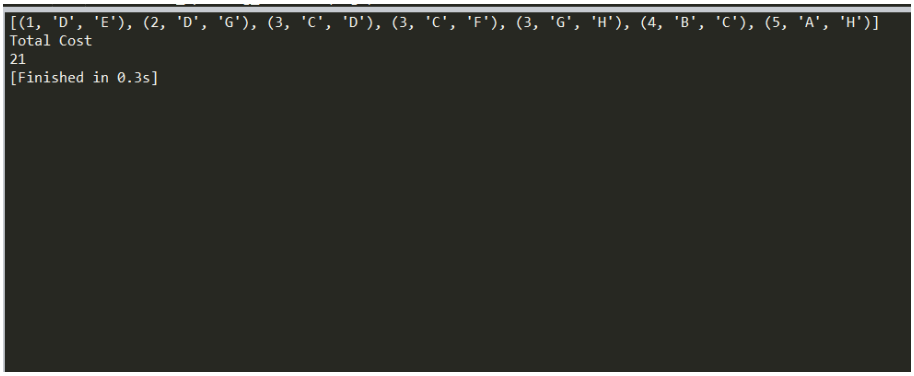

Figure 1: Time Independent Feature Set

2 Task

2.1 Procedure: Task 1

-Select the shortest edge in a network.
-Select the next shortest edge which does not create a cycle.
-Repeat step 2 until all vertices have been connected.

2.2 Procedure: Task 2

```
parent = dict()
rank = dict()

def make_set(vertice):
    parent[vertice] = vertice
    rank[vertice] = 0
    #print parent
    #print rank

def find(vertice):
    if parent[vertice] != vertice:
        parent[vertice] = find(parent[vertice])
    return parent[vertice]
```

```
def union(vertice1, vertice2):
    root1 = find(vertice1)
    root2 = find(vertice2)
    #print root1
    #print root2
    if root1 != root2:
        if rank[root1] > rank[root2]:
            parent[root2] = root1
        else:
            parent[root1] = root2
        if rank[root1] == rank[root2]: rank[root2] += 1

def kruskal(graph):
    for vertice in graph['vertices']:
        make_set(vertice)
        minimum_spanning_tree = set()
        edges = list(graph['edges'])
        edges.sort()
        global total
        total=[]
        #print edges
    for edge in edges:
        weight, vertice1, vertice2 = edge
        if find(vertice1) != find(vertice2):
            union(vertice1, vertice2)
            minimum_spanning_tree.add(edge)
            total.append(weight)

    return sorted(minimum_spanning_tree)

graph = {
'vertices': ['A', 'B', 'C', 'D', 'E', 'F', 'G','H'],
'edges': set([
(8, 'A', 'B'),
(5, 'A', 'H'),
(10, 'A', 'F'),
(4, 'B', 'C'),
(4, 'B', 'F'),
(4, 'B', 'H'),
(4, 'B', 'E'),
```

```
(3, 'C', 'F'),
(3, 'C', 'D'),
(1, 'D', 'E'),
(2, 'D', 'G'),
(6, 'D', 'F'),
(3, 'E', 'G'),
(3, 'G', 'H'),
])
}

print kruskal(graph)
print "Total Cost"
print sum (total)
```

3 Conclusion

-Kruskals algorithm is an edge based algorithm.
-Kruskal's method is more time saving.
-It solve the single-source shortest path problem.
-Prims algorithm with a heap is faster than Kruskals algorithm.