



THE UNIVERSITY
OF LAHORE
**ISLAMABAD
CAMPUS**

Artificial Intelligence (CS13217)

Lab Report

Name: Sameea Naeem
Registration #: CSU-XS16-139
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

Implementing Prim's Algorithm

Objective

To understand and implement the Prim's Algorithm

Software Tool

1. OS: Windows
2. IDE: Pycharm
3. Language: Python

1 Theory

Algorithm 1) Create a set `mstSet` that keeps track of vertices already included in MST.

2) Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.

3) While `mstSet` doesn't include all vertices

.a) Pick a vertex `u` which is not there in `mstSet` and has minimum key value.

.b) Include `u` to `mstSet`.

.c) Update key value of all adjacent vertices of `u`. To update the key values, iterate through all adjacent vertices. For every adjacent vertex `v`, if weight of edge `u-v` is less than the previous key value of `v`, update the key value as weight of `u-v`.

The idea of using key values is to pick the minimum weight edge from cut. The key values are used only for vertices which are not yet included in MST, the key value for these vertices indicate the minimum weight edges connecting them to the set of vertices included in MST.

2 Code

```
def prim(graph , root ):
```

```

assert type(graph)==dict

nodes = graph.keys()
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)

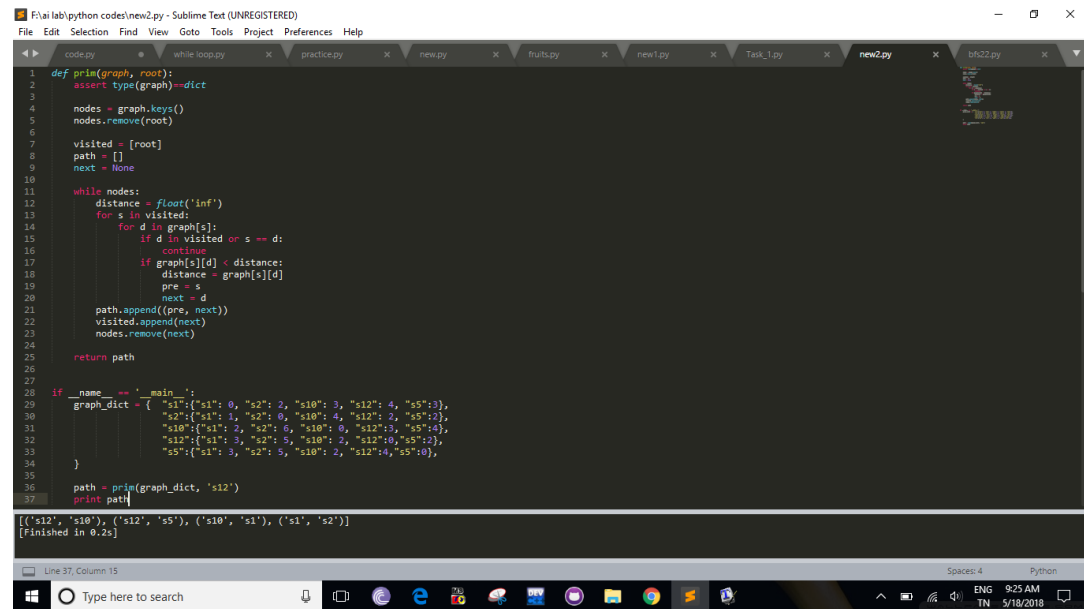
return path

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 Implement Prim's in Python language:



```
1 def prim(graph, root):
2     assert type(graph) == dict
3
4     nodes = graph.keys()
5     nodes.remove(root)
6
7     visited = [root]
8     path = []
9     next = None
10
11     while nodes:
12         distance = float('inf')
13         for s in visited:
14             for d in graph[s]:
15                 if d in visited or s == d:
16                     continue
17                 if graph[s][d] < distance:
18                     distance = graph[s][d]
19                     pre = s
20                     next = d
21             path.append((pre, next))
22             visited.append(next)
23             nodes.remove(next)
24
25     return path
26
27
28 if __name__ == '__main__':
29     graph_dict = {
30         "s1": [{"s1": 0, "s2": 2, "s10": 3, "s12": 4, "s5": 3},
31         "s2": [{"s1": 1, "s2": 0, "s10": 4, "s12": 2, "s5": 2},
32         "s10": [{"s1": 2, "s2": 0, "s10": 0, "s12": 3, "s5": 4},
33         "s12": [{"s1": 3, "s2": 5, "s10": 2, "s12": 0, "s5": 2},
34         "s5": [{"s1": 3, "s2": 5, "s10": 2, "s12": 4, "s5": 0}],
35     }
36
37     path = prim(graph_dict, 's12')
38     print path
39
40 [(['s12', 's10'], ('s12', 's5'), ('s10', 's1'), ('s1', 's2'))]
41 [Finished in 0.2s]
```

4 Conclusion:

shortest path found.