LAB 11: Dijkstra's Algorithm

CS211 – Data Structures and Algorithms
Usman Institute of Technology
Fall 2019

- **How to submit:**
  - Create an account on http://www.turnitin.com/ as a Student (if you don't have already)
  - Use following information at time of sign-up

    CS Section A
    - Class ID: 22664649
    - Enrollment Key: DSFALL19CSA

    CS Section B
    - Class ID: 22664651
    - Enrollment Key: DSFALL19CSB

A. Create a class Vertex that stores the information containing value and the distance.

1. Add a constructor of the class that takes one argument value in order to set the number of vertices. The constructor should also initialize an attribute distance and an attribute pi.

```
class Vertex:
    def __init__(self,value):
        // your code goes here
```

B. Create a class Priority Queue that implements the functions of the queue in the following order.

1. Add a constructor of the class that initializes an empty list.

```
class PriorityQueue:
    def __init__(self):
        // your code goes here
```

2. Add a function IsEmpty() which returns True if the list is empty otherwise False.

```
    def IsEmpty(self):
        // your code goes here
```

3. Add a function Enqueue() that adds Vertex object in the queue

```
    def Enqueue(self):
        // your code goes here
```

4. Add a function Dequeue() that removes the object from the queue

```
    def Dequeue(self):
        // your code goes here
```

5. Add a function ExtractMin() that returns the vertex with minimum distance from the queue.

```
    def ExtractMin(self):
        // your code goes here
```

C. Create a class DGraph that implements the functions for calculating Shortest Path in the following order.

1. Add a constructor of the class that initializes number of vertices and an adjacency matrix.

```
Class DGraph:
    def __init__(self,vertex):
        // your code goes here
```

2. Add a function AddDirectedEdges() connects the edges of source and destination.

```
    def AddDirectedEdges(self,source,dest,cost):
        // your code goes here
```

3. Add a function GetDirectedNeighbours() that returns the list of the neighbours of the source.

```
    def GetDirectedNeigbours(self,source):
        // your code goes here
```

4. Add a function DijkstraShortestPath() that calculates the shortest path for the graph.

```
    def DijkstraShortestPath(self,source):
        // your code goes here
```