**CSE 101 - Introduction to Programming**

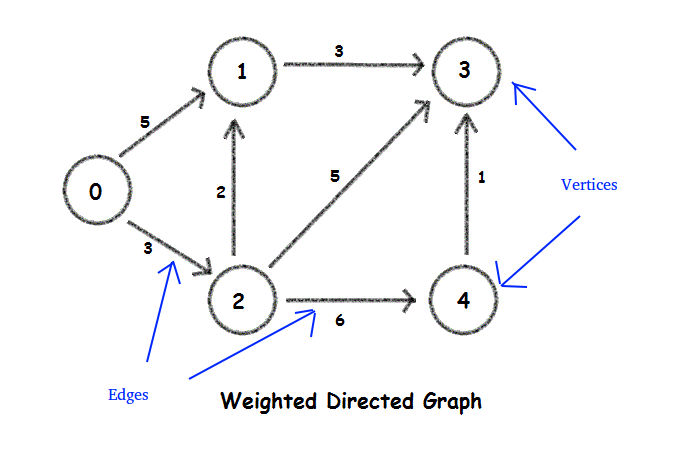
**Home Assignment 3**

**Play with Graphs**

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges.

In a graph each of the edges can have a weight and an arrow depicting the direction.  
In fig :

* Circles represent vertices. Each vertex has a name (0,1, 2,3,4)
* Arrows between vertices represent edges. (Edge between 1 --> 3)
* Each edge has a weight associated with it. (Edge between 1--> 3 has a weight =3 )

  
Figure 1: Directed weighted Graph

**Why learn graphs??**

Graphs are used to represent many real life applications. One of the simplest examples can be network of cities. Let each node represent a city and edge weight can represent the cost incurred in travelling from one city to another. Graphs are also used in social networks like linkedIn, facebook.

**In this home assignment you will be implementing some algorithm on graphs.**

**Task 1**

**Input:**

Input to the task will be a graph in the following format:

connections: [[1, 2], [3], [1, 3, 4], [], [3]]

weights: [[5, 3], [3], [2, 5, 6], [], [1]]

**For the above format input example will be:**

5

2

1 5

2 3

1

3 3

3

1 2

3 5

4 6

0

1

3 1

First line represent the number of vertex.

Next line shows the number of connection to 0th vertex i.e. 2

Next 2 line shows the connected vertex and weight of edge to 0th vertex to connected vertex and so on.

**Algorithm to Implement:**

**Algorithm 1: Dijkstra's Single source shortest path problem.**  
Given a source vertex, say vertex 1 you need to find the shortest path (the path with least weight) to all other vertices in the graph.

For eg. in Fig 1 above , If vertex 0 is the source vertex Shortest path values will be :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vertices | 1 | 2 | 3 | 4 |
| Values | 5 | 3 | 8 | 9 |

**Pseudocode:**

1. **Dijkstra(Graph,source)** // Graph will be your connections and weights
2. create vertex set Q
3. for each vertex v in Graph: // Initialization
4. dist(v) 🡨 INFINITY // unknown distance from source to v
5. prev(v) 🡨 UNDEFINED // Previous node in optimal path from source
6. add v to Q // All nodes initially in Q (unvisited node)
7. dist(source) 🡨 0 // Distance from source to source
8. While Q is not empty:
9. u 🡨 vertex in Q with min dist(u) // node with the least distance will be

selected first

1. remove u from Q
3. for each neighbour v of u: // where v is still in Q
4. alt 🡨 dist(u) + length(u, v)
5. if alt < dist(v): // A shorter path to v has been found
6. dist(v) 🡨 alt
7. prev 🡨 u
8. return dist

**Output :**

The minimum distance of each vertex from the given source vertex.

For more information you can refer the following sources :

https://brilliant.org/wiki/dijkstras-short-path-finder/

http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/

**Algorithm 2 : Breadth First Traversal /Search (BFS)**  
Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

Your task is to print the breadth first traversal of the graph given in problem 1.

Please refer the following sources for learning more on breadth first traversal and how to implement it.

https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/

https://www.tutorialspoint.com/data\_structures\_algorithms/breadth\_first\_traversal.htm

.**Task 2**

You need to again run both algorithms considering all weight to unity i.e. 1.