Computer Science 303 – Algorithms
Programming Assignment #3: Graph class
Due: 5/6/2014 by midnight

In this programming assignment, we will write a graph class that incorporates some of the graph-based algorithms we have learned in our class. The overall score of this programming assignment is 100.

Task

1. Write a graph class that can represent a directed or undirected graph. The graph class should be able to create a graph based on file input. The format of input can be either linked list or adjacency matrix. You can assume the name of a node will be a number sequentially from 1 to the total number of nodes. We will only allow user to use adjacency matrix if edges in the graph are weighted and we only allow non-negative weights in the graph.

    - For file input, the file should contain the following information in the correct order

        i. Row 1: d for directed and u for undirected

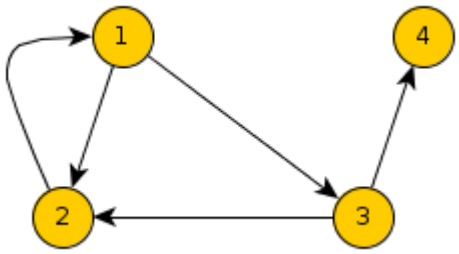
        ii. Row 2: a for adjacency matrix and l for linked list

        iii. Row 3: the number of nodes (must be >0)

        iv. The linked list or adjacency matrix

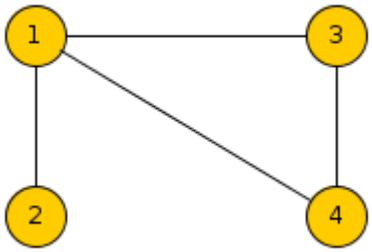
    For the format of linked list, it should be one row per node with # symbol to mark the end of that row. The last row will contain ## to indicate the end of input. For example, for a directed graph of 4 nodes, the following linked list input represents

    2 3 #
    1 #
    4 2 #
    #
    ##

If the graph is undirected, the linked list should be "symmetric" meaning that an edge will occur at both the starting node line and the ending node line. For example, we may have an undirected linked list as follows

2 3 4 #
1 #
1 4 #
1 3 #
##



If adjacency matrix format is used, the matrix for the two examples above are

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ for the digraph and $$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$ for the undirected graph.

If the graph is weighted, the ones in the adjacency matrix will be substituted as weights on the corresponding edge.

2. For the graph class, you should include appropriate member functions including at least

- Default constructor to initialize an empty graph
- One constructor to initialize an object from file. You can design this constructor by yourself.
- A copy constructor and overload the assignment operator = for the class
- Overload == and != operator for the class
- Access functions that can tell whether an edge exists in the graph by providing two nodes.

- Mutate functions that will allow user to delete a node or delete an edge between two nodes.
- A destructor for the class
- Two traversal functions that will traverse the graph using either depth first search or breath first search. Display the order of visited nodes as the output.
- A function that will calculate the minimum distance from a given a node to all other nodes in the graph. You should implement either Dijkstrar's algorithm or Bellman-ford.