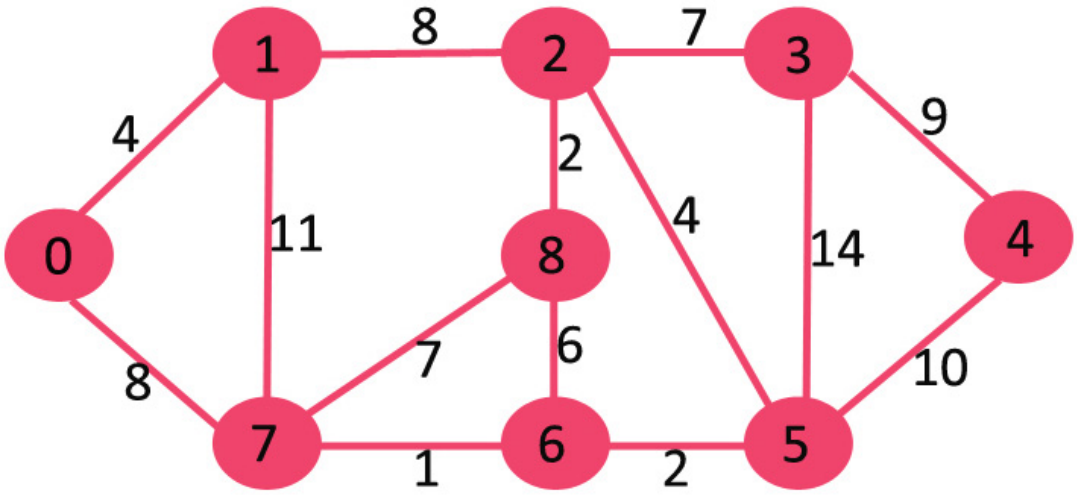
CPS112: Computational Thinking with Data Structures & Algorithms
Lab 13: On the (shortest) road again

1. Review the your notes for BFS, DFS, and Dijkstra's algorithm. Then, using vertex 0 as the starting point perform by hand. For BFS/DFS when there are multiple neighbors process them in numerical order (i.e., for vertex 2 process vertex 3 before 5 and 8)
   (a) BFS make sure to record the distance and predecessor.
   (b) DFS make sure to record the discovery and finish times
   (c) Dijkstra's algorithm. Be sure to give the final distances and previous vertex for each vertex.



– MIDWAY CHECK –

2. In *ListGraph.java* fill in the methods. For this problem you may assume the graph will always be undirected.
   (a) `addEdge`: Calls `addVertex` for both vertices, then adds the second vertex to the `ArrayList` associated with the first vertex AND adds the first vertex to the `ArrayList` associated with the second vertex.
   (b) `getNeighbors`: Returns an `ArrayList` of neighbor vertices
   (c) `isAdjacent`: Returns `true` if the two vertices are adjacent, `false` otherwise.
   Test your code using the graph from above in problem 1.

3. In *ListGraph.java* you have been given the stub of three methods, `BFS`, `DFS`, and `DFSVisit`. Based on your notes and your work from problem 1 fill in these three methods.
   (a) I have given you an updated `Vertex` class with several public instance variables to help you implement `BFS` and `DFS`.
   (b) There is a method in the `Graph` class to reset vertex properties that you can use.
   (c) I have written the `main` method to reproduce the graph above and perform the two searches so you can check the output with your solutions from problem 1.

– FINAL CHECK –