Week 10 Home Exercises Solutions

True/False

For all answers to the (T/F) question, please provide a short reason why as well.

1. To implement Dijkstra's shortest path algorithm on unweighted graphs so that it runs in linear time, the data structure to be used is stack. [T/F]

   Solution: False, The shortest path in an un-weighted graph means the smallest number of edges that must be traversed in order to reach the destination in the graph. This is the same problem as solving the weighted version where all the weights happen to be 1. If we use Queue (FIFO) instead of Priority Queue (Min Heap), we get the shortest path in linear time $O(|V| + |E|)$. Basically we do BFS traversal of the graph to get the shortest paths. So, we need to use a queue.

2. In an unweighted, undirected connected graph, the shortest path from a node S to every other node is computed most efficiently, in terms of time complexity by performing a DFS starting from S. [T/F]

   Solution: False, DFS cannot be used for finding shortest paths.

3. The time complexity of Bellman-Ford single-source shortest path algorithm on a complete graph of n vertices is $O(n^3)$.[T/F]

   Solution: True, see lecture notes.

4. If we make following changes to Dijkstra, then it can be used to find the longest simple path. Assume that the graph is acyclic. [T/F]

   (a) Initialise all distances as minus infinite instead of plus infinite.

   (b) Modify the relax condition in Dijkstra's algorithm to update distance of an adjacent vertex v of the currently considered vertex u only if $dist[u] + graph[u][v] > dist[v]$. In shortest path algorithm, the sign is opposite.

   Solution: False, In shortest path algorithm, we pick the minimum distance vertex from the set of vertices for which distance is not finalised yet. And we finalise the distance of the minimum distance vertex. For maximum distance problem, we cannot finalise the distance because there can be a longer path through not yet finalised vertices.