Student Information

Name: Wang Tianduo Student ID: 1002963

Due Date: 20 Nov, 2018 - 11:59PM.

Submit answers on eDimension in pdf format. Submission without student information will NOT be marked! Any questions regarding the homework can be directed to the TA through email (contact information on eDimension).

Week 10 Home Exercises

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]

   Essentially, Dijkstra's shortest path algorithm is breadth-first-search, and the data structure that used in BFS should be queue. If we use a stack, since it is First-in-last-out, then, the algorithm will conduct DFS instead.

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]

   DFS does not guarantee that if node 1 is visited before another node 2 starting from a source vertex, then node 1 is closer to the source than node 2

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]

   The general time complexity of Bellman-Ford algorithm is O(VE), where V is the number of vertices and E is number of edges. Also, a complete graph is a graph in which each pair of edges is connected by an edge. So complete graph is a dense graph and the number of edges is approximately equal to v^2. Thus, in this case, if there are n vertices in graph, the time complexity is O(n^3)

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.

In Dijkstra's shortest path algorithm, we pick the minimum distance vertex from the set of vertices for which distance is not finalized yet. And then we finalize the distance of the minimum distance vertex. It is a greedy strategy. But for longest distance problem, we cannot finalize the distance because there can be a longer path through not yet finalized vertices.