Multiple Choice Questions

Q1. Indicate the runtime of Dijkstra's algorithm when the implementation is based on a binary heap. (E = edges; V = vertices)

A. O(ElogV)

B. O(V2)

C. O(E + VlogV)

D. O(E + V)

Answer : A

If implemented using a binary min heap, its runtime is O(ElogV).

Q2. What is the shortest path from node A to node F?

A. A -> B -> D -> F

B. A -> C -> B -> E -> F

C. A -> F

D. A -> C -> E -> F

Answer : D

A -5-> C -2-> E -3-> F = 10

Q3. Using the graph from the previous question, if we apply Dijkstra's algorithm to find the shortest distance between node A and all the others, in what order do the nodes get included into the visited set (i.e their distances have been finalized)?

A - B C F G E D

B - B C G E F D

C - C B E F G D

D - C B E G F D

Answer : B

The nodes are added to the visited set in terms of the shortest distance to the source.

Q4. To implement Dijkstra’s shortest path algorithm on unweighted graphs so that it runs in linear time, the data structure to be used is:

A. Stack

B. Heap

C. Queue

D. Binary Tree

Answer : C

If we use Queue (FIFO) instead of Priority Queue (Min Heap), we get the shortest path in linear time O(|V| + |E|).

Q5. In a weighted, undirected graph if we apply Dijkstra's algorithm to find the shortest path between two nodes. If we add 1 to all the edge weights, does the shortest path remain the same?

A - Yes

B - No

Answer : B

If we use the graph on question 2 and increase all edge weights by 1, the shortest path from node A to node F is no longer A -> C -> E -> F, it becomes A -> F.

Q6. If all edges have the same weight in an undirected graph, which algorithm will find the shortest path between two nodes more efficiently?

A - Dijkstra

B - Bellman-Ford

C - Depth-First Search

D - Breadth-First Search

Answer : D

Breadth-First Search has time complexity O(|V| + |E|).

Q7. Dijkstra's algorithm is based on which paradigm?

A - Greedy paradigm

B - Backtracking paradigm

C - Dynamic Programming paradigm

D - Divide and Conquer paradigm

Answer : A

Dijkstra relates to the greedy approach since we select the node with the shortest distance from the set of unvisited nodes.

Q8. Given an undirected graph with negative edge weights, will Dijkstra's algorithm find the shortest path between two nodes correctly?

A - Yes

B - No

Answer : B

Dijkstra relies on the fact that all edges are non-negative, adding an edge can never make a path shorter. This is not the case with negative edge weights.

In this example if we are trying to find the shortest path between node A and node B

1. Assign D[C] = 0, D[B] = 1 and D[D] = 20.

2. We explore node C and no changes are made.

3. We explore node B and D[D] is updated to -39.

4. We explore node D.

The shortest path to B is -20 and not 1.