National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	
Program:	**BS(SE)**	**Semester:**	**6**
Duration:	**20 minutes**	**Total Marks:**	**15 marks**
Paper Date:	**15 May, 2025**	**Weight**	
Section:	**SE-A**	**Page(s):**	2
Exam Type:	**Quiz4-version2**		

Student Name: **Roll No #:**

Instructions: **Solve Question 2 on back side as well**

1) What is the worst-case time complexity of BFS in an adjacency list representation of a graph?
A. $O(V^2)$
B. $O(V + E)$
C. $O(E \log V)$
D. $O(V \log V)$

2) Which of the following edge types CANNOT exist in DFS of an undirected graph?
A. Tree edge
B. Back edge
C. Forward edge
D. Cross edge

3) What happens if you run Prim's algorithm starting from a vertex with no incident edges?
A. It returns a spanning forest
B. It returns an MST
C. It fails with an error
D. It processes all vertices

4) In a connected graph with n vertices and m edges, how many edges are not in any possible MST?
A. m - (n - 1)
B. Depends on edge weights
C. All edges with maximum weights
D. All edges not on the DFS tree

5) Prim's and Kruskal's algorithms produce the same MST if:
A. The graph is a complete graph
B. All edge weights are equal
C. The graph has unique edge weights
D. The graph is a tree

6) Which is TRUE about the DFS finishing times in a DAG?
A. All sinks finish before sources
B. All sources finish before sinks
C. Sinks finish last
D. Finishing time is irrelevant in DAGs

7) Which of the following statements about DFS is INCORRECT?
A. DFS can be used to detect cycles in a graph
B. DFS can produce topological sorting of a DAG
C. DFS tree in a disconnected graph is always connected
D. DFS discovery and finishing times are essential in articulation point detection

8) Kruskal's algorithm can be best optimized using which technique?
A. Binary Tree Sorting
B. Disjoint Set Union (Union-Find)
C. Breadth First Search
D. Segment Tree

Q2) You are given a grid of cells where each cell is either empty (0) or blocked (1). You start at position (0,0) and want to reach (n-1, m-1). You can move in four directions: up, down, left, right.

Grid:

[0, 0, 0, 0, 1]

[1, 1, 0, 1, 0]

[0, 0, 0, 1, 0]

[0, 1, 1, 0, 0]

[0, 0, 0, 0, 0]

Using DFS, determine the minimum number of steps needed to reach the destination. **Show working as well**

National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	
Program:	**BS(SE)**	**Semester:**	**6**
Duration:	**20 minutes**	**Total Marks:**	**15 marks**
Paper Date:	**15 May, 2025**	**Weight**	
Section:	**SE-A**	**Page(s):**	2
Exam Type:	**Quiz4-version1**		

Student Name: **Roll No #:**

Instructions: **Solve Question 2 on back side as well**

1) Which of the following statements is TRUE regarding the BFS traversal of a directed graph?
A. BFS always visits vertices in lexicographical order
B. BFS tree edges never form a cycle
C. Cross edges in BFS traversal imply a back edge
D. BFS guarantees shortest paths even in weighted graphs

2) Given a graph with negative weight edges but no negative cycles, which traversal is appropriate for finding shortest path from a source?
A. DFS
B. BFS
C. Prim's
D. None of the above

3) Which statement is TRUE about BFS vs DFS?
A. DFS uses more memory than BFS in sparse graphs
B. BFS can get stuck in cycles without visited array
C. DFS is guaranteed to find the shortest path
D. BFS is unsuitable for cycle detection

4) In a connected graph with n vertices and m edges, how many edges are not in any possible MST?
A. m - (n - 1)
B. Depends on edge weights
C. All edges with maximum weights
D. All edges not on the DFS tree

5) Prim's and Kruskal's algorithms produce the same MST if:
A. The graph is a complete graph
B. All edge weights are equal
C. The graph has unique edge weights
D. The graph is a tree

6) Let G be an undirected graph. Suppose we run DFS on G, and classify edges as tree, back, forward, or cross edges. Which of the following must be TRUE?
A. Back edges always form a cycle
B. Cross edges can appear in DFS tree of an undirected graph
C. Every forward edge implies a cycle
D. Cross edges never exist in undirected graphs

7) Which of the following statements about DFS is INCORRECT?
A. DFS can be used to detect cycles in a graph
B. DFS can produce topological sorting of a DAG
C. DFS tree in a disconnected graph is always connected
D. DFS discovery and finishing times are essential in articulation point detection

8) Which of the following guarantees a unique MST in a connected, weighted undirected graph?
A. All edge weights are even numbers
B. All edge weights are unique
C. The graph has no cycles
D. The number of vertices is prime

Q2) You are given a grid of cells where each cell is either empty (0) or blocked (1). You start at position (0,0) and want to reach (n-1, m-1). You can move in four directions: up, down, left, right.

Grid:

[0, 0, 0, 0, 1]

[1, 1, 0, 1, 0]

[0, 0, 0, 1, 0]

[0, 1, 1, 0, 0]

[0, 0, 0, 0, 0]

Using BFS, determine the minimum number of steps needed to reach the destination. **Show working as well**

National University of Computer and Emerging Sciences
Lahore Campus

	Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	
	Program:	**BS(SE)**	**Semester:**	**6**
	Duration:	**20 minutes**	**Total Marks:**	**15**
	Paper Date:	**15 May, 2025**	**Weight**	
	Section:	**SE-6A**	**Page(s):**	
	Exam Type:	**Quiz5-version1**		

Student Name: **Roll No #:**

Instructions: **Solve Question 2 on back side as well**

1) **What happens if Dijkstra's algorithm is applied to a graph with a negative edge?**
A. It returns the shortest path
B. It returns an incorrect result
C. It crashes
D. It performs slower but correctly

2) **Which of the following is an application of Bellman-Ford?**
A. Topological sorting
B. Cycle detection in undirected graphs
C. Internet routing protocols (e.g., RIP)
D. Minimum spanning tree construction

3) **In Bellman-Ford algorithm, how many times are all edges relaxed in the worst case?**
A. |V|
B. |V| - 1
C. |E|
D. log|V|

4) **Which of the following can be used for shortest paths in a graph with all edge weights equal to 1?**
A. Dijkstra's
B. DFS
C. BFS
D. Kruskal's

5) **In Dijkstra's algorithm, which structure ensures minimum distance vertex is always chosen next?**
A. Stack
B. Queue
C. Min Heap / Priority Queue
D. Set

6) **In a weighted graph, the output of Bellman-Ford is always correct if:**
A. There are no self-loops
B. The graph has no cycles
C. There are no negative weight cycles
D. The graph is a tree

7) **What indicates the presence of a negative weight cycle in Bellman-Ford?**
A. Distances don't change in the final iteration
B. Some distances become zero
C. A vertex is visited more than once
D. Relaxation still changes a distance in the |V|th iteration

8) **What is the major drawback of the Bellman-Ford algorithm compared to Dijkstra?**
A. It doesn't work on sparse graphs
B. It cannot detect negative cycles
C. Its time complexity is higher
D. It only works on trees

```
   5       1
 A -----> B ----> C
 |       ^       ^
 |       |       |
 |       |       |
1|      3|       | 6
 |       |       |
 v       |       |
 D -----> E ----> F
   2       1
```

Q2: Start at vertex A and use Dijkstra's algorithm to find the shortest path to all other vertices**. Show each iteration of the algorithm.**

National University of Computer and Emerging Sciences
Lahore Campus

	Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	
	Program:	**BS(SE)**	**Semester:**	**6**
	Duration:	**20 minutes**	**Total Marks:**	**15**
	Paper Date:	**15 May, 2025**	**Weight**	
	Section:	**SE-6A**	**Page(s):**	
	Exam Type:	**Quiz5-version2**		

Student Name: **Roll No #:**

Instructions: **Solve Question 2 on back side as well**

1) **Which of the following is FALSE about Dijkstra's algorithm?**
A. It can be used to compute SPT
B. It fails with negative edge weights
C. It uses a greedy strategy
D. It can detect negative cycles

2) **In Bellman-Ford algorithm, how many times are all edges relaxed in the worst case?**
A. |V|
B. |V| - 1
C. |E|
D. log|V|

3) **When does Bellman-Ford give better performance than Dijkstra's algorithm?**
A. On sparse graphs with only positive weights
B. On graphs with dense connectivity
C. On graphs with negative weights and sparse structure
D. On graphs with uniform weights

4) **In a weighted graph, the output of Bellman-Ford is always correct if:**
A. There are no self-loops
B. The graph has no cycles
C. There are no negative weight cycles
D. The graph is a tree

5) **Suppose Bellman-Ford runs for |V| iterations and still changes a distance value in the last iteration. What does this indicate?**
A. The graph is a tree
B. The graph is disconnected
C. There exists a negative weight cycle
D. The shortest path hasn't converged

6)) **Which of the following can be used for shortest paths in a graph with all edge weights equal to 1?**
A. Dijkstra's
B. DFS
C. BFS
D. Kruskal's

7) **Which of the following graphs would cause Dijkstra's algorithm to fail to find the correct shortest paths?**
A. Graph with no cycles
B. Graph with disconnected vertices
C. Graph with negative weight edges
D. Graph with uniform weights

8) **You are given a directed graph with negative edge weights but no negative cycles. Which algorithm guarantees correct shortest path results from a single source?**
A. Dijkstra's
B. BFS
C. Bellman-Ford
D. DFS

```
    3      4
 A -----> B ----> C
 |        ^       ^
 |        |       |
 |        |       |
2|       5|       | 4
 |        |       |
 v        |       |
 D -----> E ----> F
    1      2
```

Q2: Start at vertex A and use Dijkstra's algorithm to find the shortest path to all other vertices**. Show each iteration of the algorithm.**

National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	
Program:	**BS(SE)**	**Semester:**	**6**
Duration:		**Total Marks:**	**20**
Paper Date:	**14 May, 2025**	**Weight**	
Section:	**SE-B**	**Page(s):**	
Exam Type:	**Quiz4-version2**		

Student Name: ______________________ **Roll No #:** ____________

Section:

Instructions: **Solve Question 2 on back side and Write Pseudo Code only**

1)**If the heaviest edge in an MST is replaced with a lighter edge not in the original graph, what happens?**
A) The MST remains unchanged
B) A new MST must be formed
C) The graph becomes disconnected
D) It depends on the edge's position

2)**Given edges sorted by weight: (1,2)=2, (2,3)=3, (3,4)=4, (1,4)=5. What is the MST total weight?**
A) 9
B) 10
C) 11
D) 12

3)**What is the worst-case space complexity of BFS in a balanced binary tree with height h?**
A) O(h)
B) O(2^h)
C) $O(h^2)$
D) O(log h)

4)**Which step determines Prim's improved time complexity with a Fibonacci heap?**
A) Extracting the minimum edge
B) Decreasing key operations
C) Union-Find operations
D) Sorting edges

5)**Adding an edge to a graph with a unique MST. When does the MST weight decrease?**
A) Always
B) If the new edge is lighter than all in the MST
C) If the new edge creates a cycle with a heavier edge
D) Never

6)**In a disconnected undirected graph with 10 nodes and 3 components, how many times does BFS restart?**
A) 1
B) 3
C) 10
D) 7

7)**A graph has all edges with weight 5. How many distinct MSTs exist if the graph is complete with 4 nodes?**
A) 1
B) 4
C) 16
D) 24

8)**Bidirectional BFS is most effective when:**
A) The graph is unweighted
B) The goal is to find all connected components
C) The target node is known, and the graph branches exponentially
D) The graph is a tree

9)**A: [B,C] B: [D] C: [E] D: [C] E: [], What is the finishing order (from first finished to last) if you run a recursive DFS starting at A and visiting neighbors in alphabetical order?**
A. A, B, D, C, E
B. E, C, D, B, A
C. D, B, E, C, A
D. B, D, E, C, A

10)**On the same graph, which of the following is a valid topological sort?**
A. A, B, D, C, E
B. A, C, E, B, D
C. B, A, D, C, E
D. D, B, A, C, E

Q2. You are given an m x n grid where each cell contains one of the following characters: 'U' (up), 'D' (down), 'L' (left), or 'R' (right). Each cell (i, j) has a directed edge to another cell based on its direction. For example, a cell with 'U' points to (i-1, j). If the edge leads outside the grid, the cell has no outgoing edge.
Return the number of distinct cycles in the grid.
grid = [

["R", "D"],
["L", "U"]
], (0,0) → (0,1) → (1,1) → (1,0) → (0,0) → cycle = 1

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Design and Analysis of Algorithms	Course Code:	
	Program:	BS(SE)	Semester:	6
	Duration:		Total Marks:	20
	Paper Date:	14 May, 2025	Weight	
	Section:	SE-B	Page(s):	
	Exam Type:	Quiz5-version1		

Student Name: **Roll No #:** **Section:**

Instructions: **Solve Question 2 on back side and Write Pseudo Code only**

1)**Edges: X→Y(6), X→Z(2), Z→Y(–3), Y→W(4), Z→W(7)**
Q: What's the true shortest distance X→W?
A) 3
B) 6
C) 7
D) 9

2)**In a dense graph (E ≈ V^2), which implementation of Dijkstra (array vs. binary-heap) asymptotically approaches the same running time?**
A) Array & heap both O(V^2)
B) Array O(V^2), heap O(E log V)
C) Array O(V log V), heap O(E log V)
D) Array O(E), heap O(V^2)

3)**Edges: P→Q(3), Q→R(–4), R→S(2), S→Q(1), P→S(10)**
Q: Running Bellman–Ford from P, on which pass do you first detect a negative-cycle relaxation?
A) 1st
B) 2nd
C) 3rd
D) never (no negative cycle)

4)**You're modeling flights with possible layovers, some of which offer time credits (negative layover time).Which approach finds the fastest itinerary and detects impossible loops?**
A) Pure Dijkstra
B) Bellman–Ford + cycle check
C) A* with zero heuristic
D) Bidirectional BFS

5)**In which graph-data scenario is Johnson's algorithm (Bellman–Ford + Dijkstra) strictly preferable?**
A) Single-source only
B) All-pairs on a sparse graph with negative edges
C) Dense graph with positives only
D) Dynamic streaming edges

6)**M→N(2), M→O(5), N→O(–3), O→P(4), N→P(10)**
What's the sequence of relaxations Bellman–Ford will perform on pass 2 to settle M→P?
A) M→N, N→O, O→P
B) M→O, O→P, N→P
C) N→O, O→P, N→P
D) M→N, N→P, O→P

7)**You have a graph where all edges are non-negative, but one edge weight is vastly larger than the others. Which behavior may degrade Dijkstra's performance in practice?**
A) Too many heap operations on small keys
B) Numeric overflow on large weights
C) Repeated extraction of a far-away node
D) Starvation of nodes with medium weights

8)**Which of these modifications can make Dijkstra correctly handle one negative edge without redesigning it entirely?**
A)Run Bellman–Ford only on that edge
B) Temporarily increase all weights by a constant
C) Reverse the graph and rerun
D) None—any negative edge breaks its greedy guarantee

9)**Road network augmented with live traffic delays (only increases, never negative). Which dynamic approach efficiently updates shortest paths without full recompute?**
A) Re-run Dijkstra from scratch
B) Delta-stepping variations of Dijkstra
C) Bellman–Ford incremental
D) Floyd–Warshall precompute

10)**How many full-edge-relaxation passes does Bellman–Ford make to compute shortest paths (before the cycle-check pass)?**
A) V
B) V–1
C) E
D) V+1

Q2. You are given a directed graph G = (V, E) with non-negative edge weights. Each edge represents a road between two cities, and the weight corresponds to the toll cost for that road. You are also given a source city s ∈ V. **Design an algorithm to compute the minimum total toll cost from the source city s to every other city in the graph, you can use a toll waiver on at most one edge in each path(algorithmically decide when to use the waiver). The toll waiver allows you to travel on one road without paying its toll, reducing that edge's cost to zero. This strategic use of the waiver can significantly impact the total cost of the journey.**
Cities: 1, 2, 3, 4
Roads:[(1, 2) with a toll cost of 10, (2, 3) with a toll cost of 10 ,(1, 3) with a toll cost of 50 ,(3, 4) with a toll cost of 10]
Path: 1 → 2 → 3 → 4 <u>Without using the toll waiver</u>:{1 to 2: 10} {2 to 3: 10} {3 to 4: 10} Total Cost: 10 + 10 + 10 = 30
<u>Using the toll waiver on edge (3, 4)</u>: {1 to 2: 10} {2 to 3: 10} {3 to 4: 0 (toll waived)} Total Cost: 10 + 10 + 0 = 20

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	
	Program:	**BS(SE)**	**Semester:**	**6**
	Duration:		**Total Marks:**	**20**
	Paper Date:	**14 May, 2025**	**Weight**	
	Section:	**SE-B**	**Page(s):**	
	Exam Type:	**Quiz5-version2**		

Student Name: **Roll No #:** **Section:**

Instructions: **Solve Question 2 on back side and Write Pseudo Code only**

1)**Edges: A→B(4), A→C(2), B→D(7), C→D(3), D→E(1)**
What's the shortest distance A→E, and via which intermediate?
A) 7 via A–B–D–E
B) 6 via A–C–D–E
C) 8 via A–B–D–E
D) 9 via A–C–D–E

2)**Edges: S→P(5), S→Q(2), P→R(–3), Q→R(4), R→T(3)**
Bellman–Ford settles S→T on which pass?
A) 1st
B) 2nd
C) 3rd
D) never

3)**Edges: X→Y(3), Y→Z(5), X→Z(10), Z→W(–4), Y→W(2)**
After running Dijkstra, what's the tentative X→W before relaxing Z→W?
A) 7
B) 8
C) 9
D) 10

4)**Dijkstra with a Fibonacci heap runs in O(E + V log V). For a sparse graph (E≈V), this is equivalent to which complexity?**
A) O(V log V)
B) O(V^2)
C) O(E log V)
D) O(E^2)

5)**Why does Bellman–Ford use exactly V–1 full edge-relaxation passes to compute shortest paths?**
A) To detect cycles
B) To propagate distances across the longest simple path of length V–1
C) Because of priority-queue limitations
D) Arbitrary but proven bound

6)**Which SSSP algorithm is best for a directed acyclic graph (DAG)?**
A) Dijkstra
B) Bellman–Ford
C) Topological-order relaxation
D) Floyd–Warshall

7)**A ride-sharing app models trips as graph edges, with surge pricing sometimes negative (discount).Which algorithm finds the cheapest route and flags impossible discount loops?**
A) Dijkstra
B) Bellman–Ford
C) A* with zero heuristic
D) Yen's k-shortest paths

8)**Live traffic updates only increase travel-time weights on edges. What dynamic technique updates shortest paths fastest without full recompute?**
A) Re-run Dijkstra from scratch
B) Delta-stepping incremental Dijkstra
C) Frequent Bellman–Ford passes
D) Precompute all-pairs and lookup

9)**Johnson's algorithm combines Bellman–Ford with Dijkstra to compute all-pairs shortest paths on sparse graphs with negative edges. What's its time complexity?**
A) O(VE + V E log V)
B) O(V E + V E log V)
C) O(VE + V E log V)
D) O(V E + E log V)

10)**You have one negative-weight edge but no cycles. Which single change lets you still use Dijkstra correctly?**
A) Add a large constant to all edges
B) Reverse edge directions
C) Preprocess that one edge with Bellman–Ford, then run Dijkstra
D) No change—Dijkstra fails with any negative edge

Q2.You are navigating a **directed weighted graph** representing a city map, where edges indicate roads and weights represent travel times in minutes. You start from a source node S.You are allowed to use a **one-time speed boost**, which lets you traverse **one edge** at **half its travel time** (rounded down if needed). The boost can be used **once**, and only in the forward direction. Return an array of minimum travel times from the source node s to every other node, using the speed boost on at most one edge in each path. **Graph : [1 → 2 (8) 2 → 3 (4) 1 → 3 (20)]**
Source: 1 Path 1 → 2 → 3 = 8 + 4 = 12
Use boost on 1 → 2: 4 + 4 = 8(best) Use boost on 2 → 3: 8 + 2 = 10 Use boost on 1 → 3: 10

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	
	Program:	**BS(SE)**	**Semester:**	**6**
	Duration:		**Total Marks:**	**20**
	Paper Date:	**14 May, 2025**	**Weight**	
	Section:	**SE-B**	**Page(s):**	
	Exam Type:	**Quiz4-version1**		

Student Name: **Roll No #:** **Section:**

Instructions: **Solve Question 2 on back side and Write Pseudo Code only**

1)**A: [B,C] B: [D] C: [E] D: [C] E: [], What is the finishing order (from first finished to last) if you run a recursive DFS starting at A and visiting neighbors in alphabetical order?**
A. A, B, D, C, E
B. E, C, D, B, A
C. D, B, E, C, A
D. B, D, E, C, A

2)**On the same graph, which of the following is a valid topological sort?**
A. A, B, D, C, E
B. A, C, E, B, D
C. B, A, D, C, E
D. D, B, A, C, E

3)**You want to preprocess a static weighted graph for many MST queries after deleting one edge each time. Which gives fastest per-query time?**
A. Run Kruskal from scratch each time: O(E log V)
B. Precompute MST and maintain dynamic tree structure: O(log V) per delete
C. Maintain adjacency matrix and recompute BFS: $O(V^2)$
D. Use Prim's with binary heap each time: O(E log V)

4)**You run BFS and DFS starting at node 1 on the same undirected graph. Which can produce the same spanning tree?**
A. Always
B. Never
C. Sometimes, if the graph is a tree
D. Only if the graph is complete

5)**A graph's edge weights are all distinct. Which yields a unique MST?**
A. Kruskal's
B. Prim's
C. Borůvka's
D. All of the above

6)**Which statement is false for a BFS tree rooted at s in an unweighted graph?**
A. It's a spanning tree of s's reachable vertices
B. The level of each vertex equals its distance from s
C. Every edge in the original graph connects vertices whose levels differ by at most 1
D. It minimizes the sum of distances in the tree

7)**You need to find the number of connected components in an undirected graph given as an edge list. Which is the fastest approach?**
A. Repeated BFS from unvisited nodes
B. Repeated DFS from unvisited nodes
C. Union-find on all edges, then count roots
D. Construct MST and count trees

8)**After Kruskal's, each disconnected component yields a:**
A) Hamiltonian cycle
B) Minimum spanning forest
C) DFS tree
D) Shortest-path tree

9)**To update an MST after deleting one edge, best worst-case per-query time uses:**
A) Kruskal from scratch O(E log V)
B) Dynamic-tree structure O(log V)
C) Re-run Prim's O(E log V)
D) Adjacency-matrix DFS $O(V^2)$

10)**Given $E = \Theta(V^2)$, Prim's with binary heap runs in:**
A) $O(V^2)$
B) O(E + V log V)
C) O(E log V)
D) O(V + E)

Q2. You are given an m x n grid where each cell contains one of the following characters: 'U' (up), 'D' (down), 'L' (left), or 'R' (right). Each cell (i, j) has a directed edge to another cell based on its direction. For example, a cell with 'U' points to (i-1, j). If the edge leads outside the grid, the cell has no outgoing edge.

Cells are part of the same connected component if you can reach them from one another by following directions (directed path).

Return the number of connected components in the grid. grid = [
["R", "L"],
["R", "L"]

], CC = 2 , → (0,0) “R” → (0,1) → (0,1) “L” → (0,0)