Meridian University

Department of Computer Science

CS 301: Algorithms & Data Structures

Final Examination · Fall Term 2026

✔ **SOLUTION KEY**

Instructor: Dr. Elena Whitfield

Duration: 120 minutes

Total Points: 100

Term: Fall 2026

Materials: One handwritten note card

Date: December 11, 2026

Instructions

- Write all answers in the space provided; overflow work may continue on the back of the page if clearly labeled.
- Show complete work for Parts III and IV — unsupported final answers receive at most half credit.
- Calculators are permitted; network-connected devices are **not**.
- Once the proctor calls time, put down all writing instruments immediately.
- Academic integrity: this exam is governed by the Meridian University Honor Code (Section 4.2). Submitting this exam constitutes an affirmation that you neither gave nor received unauthorized assistance.

Part	Topic	Points	Score
I	Multiple Choice	10	—
II	Short Answer	20	—
III	Problem Solving & Proofs	30	—
IV	Case Study: Data-Center Routing	40	—
	Total	**100**	—

Part I — Multiple Choice — 10 points

Question 1. What is the worst-case time complexity of binary search on a sorted array of n elements? [2 points]

(A) $O(n)$

(B) $O(n \log n)$

(C) $O(\log n)$

(D) $O(1)$

✔ **Correct answer: (C)** Each comparison halves the remaining search space, giving $T(n) = T(n/2) + O(1)$, which solves to $O(\log n)$.

Question 2. Which data structure is most appropriate for implementing a breadth-first traversal of a graph? [2 points]

(A) Stack

(B) Queue

(C) Priority queue

(D) Doubly linked list

✔ **Correct answer: (B)** BFS visits nodes in the order they are discovered, which is exactly FIFO behavior — a queue.

Question 3. A hash table with n entries and a table size of m has load factor $\alpha = n/m$. Under separate chaining with a well-distributed hash function, the expected time for a successful search is: [2 points]

(A) $O(1)$, independent of α

(B) $O(1 + \alpha)$

(C) $O(\log n)$

(D) $O(n)$

✔ **Correct answer: (B)** Expected chain length is α, plus $O(1)$ for computing the hash and locating the bucket.

Question 4. Which of the following sorting algorithms is *not* comparison-based? [2 points]

(A) Merge sort

(B) Quicksort

(C) Heapsort

(D) Radix sort

✔ **Correct answer: (D)** Radix sort distributes elements by digit/key buckets and never compares two keys directly, so it is not bound by the $\Omega(n \log n)$ comparison-sort lower bound.

Question 5. For a directed acyclic graph with V vertices and E edges, topological sort via depth-first search runs in: [2 points]

(A) $O(V^2)$

(B) $O(V + E)$

(C) $O(E \log V)$

(D) $O(V \cdot E)$

✔ **Correct answer: (B)** DFS visits every vertex once and traverses every edge once, so the total work is linear in the size of the graph, $O(V + E)$.

Part II — Short Answer — 20 points

Question 6. Define *amortized analysis* and give one concrete example of an operation whose amortized cost differs from its worst-case cost. [5 points]

✔ **Solution.** Amortized analysis bounds the *average* cost per operation over a worst-case sequence of operations, even though individual operations may occasionally be expensive. Example: dynamic array `push_back`. A single insertion that triggers a resize costs $O(n)$, but because resizes double the capacity, the amortized cost per insertion averaged over n pushes is $O(1)$.

Question 7. Explain why a balanced binary search tree (e.g., an AVL or red-black tree) guarantees $O(\log n)$ height, while an unbalanced BST can degrade to $O(n)$ height. [5 points]

✔ **Solution.** A balanced BST enforces a structural invariant after every insertion/deletion (height-balance for AVL, color/black-height rules for red-black trees) that bounds the ratio between the shortest and longest root-to-leaf paths, forcing height to grow logarithmically with n. An unbalanced BST has no such invariant: inserting keys in sorted order degenerates the tree into a linked list of height $n - 1$.

Question 8. State the recurrence relation for merge sort and solve it using the Master Theorem. [5 points]

✔ **Solution.** $T(n) = 2T(n/2) + O(n)$. With $a = 2$, $b = 2$, $f(n) = O(n)$, we compare $n^{\log_b a} = n^{\log_2 2} = n^1$ against $f(n) = O(n)$: they match (Case 2 of the Master Theorem), so $T(n) = O(n \log n)$.

Question 9. What is the difference between a greedy algorithm and dynamic programming, and under what condition does a greedy choice still yield a globally optimal solution? [5 points]

✔ **Solution.** Dynamic programming explores overlapping subproblems and combines optimal sub-solutions, often considering multiple choices at each step and caching results. A greedy algorithm commits to a single locally optimal choice at each step without reconsidering it. A greedy strategy yields a globally optimal solution when the problem exhibits the *greedy-choice property* (a locally optimal choice is always part of some globally optimal solution) together with *optimal substructure*.

Part III — Problem Solving & Proofs — 30 points

Question 10. Prove by induction that the sum of the first n positive integers is $\dfrac{n(n+1)}{2}$. [15 points]

✔ **Solution.** **Base case** ($n = 1$): $\sum_{i=1}^{1} i = 1 = \dfrac{1(1+1)}{2}$. True.

Inductive step: Assume $\sum_{i=1}^{k} i = \dfrac{k(k+1)}{2}$ for some $k \geq 1$. Then

$$\sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1) = \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}.$$

This matches the formula at $n = k+1$, so by the principle of mathematical induction the statement holds for all $n \geq 1$. ■

Question 11. Write pseudocode for a function `isBalanced(root)` that determines whether a binary tree is height-balanced (the heights of the two subtrees of every node differ by at most 1), and state its time complexity. [15 points]

✔ **Solution.**

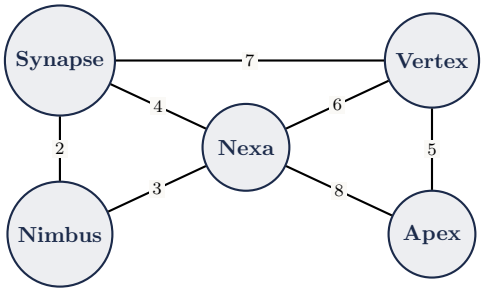
```
function height(node):
if node is null:  return 0
lh = height(node.left)
rh = height(node.right)
if lh == -1 or rh == -1 or |lh - rh| > 1:  return -1
return 1 + max(lh, rh)
function isBalanced(root):
return height(root) ≠ -1
```

Each node is visited exactly once and the -1 sentinel propagates imbalance upward without recomputing subtree heights, giving $O(n)$ time instead of the naive $O(n^2)$.

Part IV — Case Study: Data-Center Routing — 40 points

Scenario

Nexa Cloud Systems operates five regional data centers — **Nexa** (primary hub), **Synapse**, **Vertex**, **Nimbus**, and **Apex** — connected by dedicated fiber links. Link latency (in milliseconds) is shown as edge weight in the network graph below. The network operations team needs to route a replication job from **Nimbus** to **Apex** along the lowest-latency path.



Question 12. Model the network as a weighted graph. Write the adjacency matrix for the five data centers (order: Nexa, Synapse, Vertex, Nimbus, Apex), using ∞ for absent links. [10 points]

✔ Solution.

	Nexa	Syn	Vtx	Nim	Apx
Nexa	0	4	6	3	8
Syn	4	0	7	2	∞
Vtx	6	7	0	∞	5
Nim	3	2	∞	0	∞
Apx	8	∞	5	∞	0

Question 13. Apply Dijkstra's algorithm starting from **Nimbus** to find the shortest-latency path to **Apex**. Show the running distance table after each vertex is finalized. [15 points]

✔ Solution.

Step	Nexa	Synapse	Vertex	Nimbus	Apex
Init	∞	∞	∞	0	∞
Finalize Nimbus (0)	3	2	∞	—	∞
Finalize Synapse (2)	3	—	9	—	∞
Finalize Nexa (3)	—	—	9	—	11
Finalize Vertex (9)	—	—	—	—	11

Shortest path: **Nimbus** → **Synapse** → **Nexa** → **Apex** with total latency **11 ms** (via Nexa is cheaper than the direct Vertex–Apex leg, $9 + 5 = 14$).

Question 14. The operations team is deciding between an adjacency matrix and an adjacency list representation for this network as it scales to $V = 500$ data centers with $E \approx 1500$ links. Justify, with complexity analysis, which representation — and which shortest-path algorithm variant — they should choose. [15 points]

✔ Solution. An adjacency matrix costs $O(V^2)$ space and makes Dijkstra's edge-relaxation step $O(V^2)$ overall, which is wasteful for a *sparse* graph ($E \ll V^2$; here $1500 \ll 500^2 = 250{,}000$). An adjacency list costs $O(V + E)$ space, and Dijkstra with a binary min-heap runs in $O((V + E) \log V)$, which for these parameters is dramatically faster than $O(V^2)$. **Recommendation:** adjacency list with a heap-based (priority-queue) Dijkstra implementation.

Answer Key — Quick Reference

Q	Answer	Q	Answer
I.1	C	III.1	Proof (see above)
I.2	B	III.2	$O(n)$ pseudocode
I.3	B	IV.1	Adjacency matrix (see above)
I.4	D	IV.2	Nimbus–Synapse–Nexa–Apex, 11 ms
I.5	B	IV.3	Adjacency list + heap Dijkstra
II.1–II.4	See worked solutions		

— End of examination. Please verify you have answered all four parts before submitting. —