Data Structure, Quiz 2 (2026)

A. Single Choice Questions (5% each, 60%)

(1) Which change would most likely break the three laws of recursion?

(A) Adding a direct answer for the smallest input.

(B) Making the recursive call work on a smaller input.

(C) Making the recursive call use exactly the same input as the current call.

(D) Combining the result of a smaller subproblem with the current item.

(C)

A recursive algorithm needs progress toward a base case. Calling the same input again prevents that progress.

(2) Selection sort is used to sort this list in increasing order:

`[42, 17, 29, 8, 35]`
What is the list after the first two passes?

(A) `[8, 17, 29, 42, 35]`

(B) `[8, 17, 29, 35, 42]`

(C) `[17, 29, 8, 35, 42]`

(D) `[8, 29, 17, 42, 35]`

(A) or (B)

Both (A) and (B) are valid, depending on the variant used. Variant 1 — find min, swap to front → Answer (A)

- Pass 1: min is 8, swap with position 0 → [8, 17, 29, 42, 35]
- Pass 2: 17 already in place, no swap → [8, 17, 29, 42, 35]

Variant 2 — find max, swap to end (PSADS textbook) → Answer (B)

- Pass 1: max is 42, swap with position 4 → [35, 17, 29, 8, 42]
- Pass 2: max of first 4 is 35, swap with position 3 → [8, 17, 29, 35, 42]

(3) Suppose binary search is applied to the sorted list

`[2, 6, 9, 14, 18, 21, 27, 31, 35, 42, 50]`
to search for `30`. The midpoint is chosen by integer floor division. Which sequence of values is compared?

(A) `21, 35, 27, 31`

(B) `21, 31, 27, 35`

(C) `18, 27, 31, 35`

(D) `21, 35, 31`

(A)

The midpoints compare 21, then 35, then 27, then 31 before the search range becomes empty.

(4) A stable sorting algorithm sorts the following records by the second value:

```
[("B", 2), ("A", 1), ("C", 2), ("D", 1)]
```

Which result preserves stability?

(A) `[("D", 1), ("A", 1), ("C", 2), ("B", 2)]`

(B) `[("A", 1), ("D", 1), ("B", 2), ("C", 2)]`

(C) `[("A", 1), ("D", 1), ("C", 2), ("B", 2)]`

(D) `[("B", 2), ("C", 2), ("A", 1), ("D", 1)]`

(B)

Stable sorting keeps equal-key records in their original relative order: A before D for key 1, and B before C for key 2.

(5) Which statement correctly compares separate chaining and open addressing for hash-table collision resolution?

(A) Separate chaining stores colliding items outside the table slot, while open addressing searches for another slot in the table.

(B) Open addressing always uses a linked list at each slot.

(C) Separate chaining requires the table size to be a prime number, but open addressing does not.

(D) Both methods require all colliding keys to be discarded.

(A)

Chaining keeps a collection such as a list at a slot; open addressing probes other slots in the table.

(6) Which statement about quicksort using the median-of-three pivot idea is TRUE?

(A) It guarantees worst-case `O(n log n)` time for every input.

(B) It chooses a pivot from three candidate positions to reduce the chance of a very poor split.

(C) It turns quicksort into a stable sorting algorithm.

(D) It removes the partitioning step from quicksort.

(B)

Median-of-three is a pivot-selection heuristic. It often improves partition quality but does not guarantee the worst case.

(7) For a sparse graph with many vertices and relatively few edges, which representation is usually more space efficient?

(A) Adjacency matrix, because it stores only existing edges.

(B) Adjacency list, because it stores each vertex's actual neighbors rather than every possible pair.

(C) Both always use exactly the same amount of memory.

(D) A stack, because it stores all edges in sorted order.

(B)

An adjacency matrix uses space for all vertex pairs; an adjacency list mostly stores existing adjacency relationships.

(8) The following is the adjacency-list representation of an undirected graph. Each row lists the neighbors of one vertex. Neighbors are processed in alphabetical order.

```
A: B, C
B: A, D, E
C: A, F
D: B
E: B, F
F: C, E
```

What is the BFS visitation order starting from `A`?

(A) `A, B, D, E, F, C`

(B) `A, C, F, E, B, D`

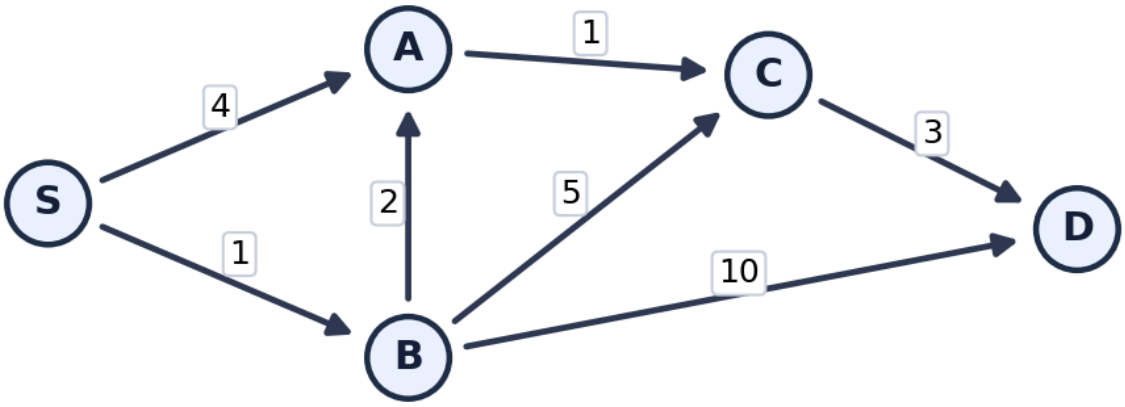
(C) `A, B, C, D, E, F`

(D) `A, B, C, F, E, D`

(C)

BFS visits all distance-1 neighbors B and C before distance-2 vertices D, E, and F.

(9) Consider the following directed weighted graph with nonnegative edge weights.



Using Dijkstra's algorithm from `S`, what is the order in which vertices are finalized?

(A) `S, A, B, C, D`

(B) `S, B, A, C, D`

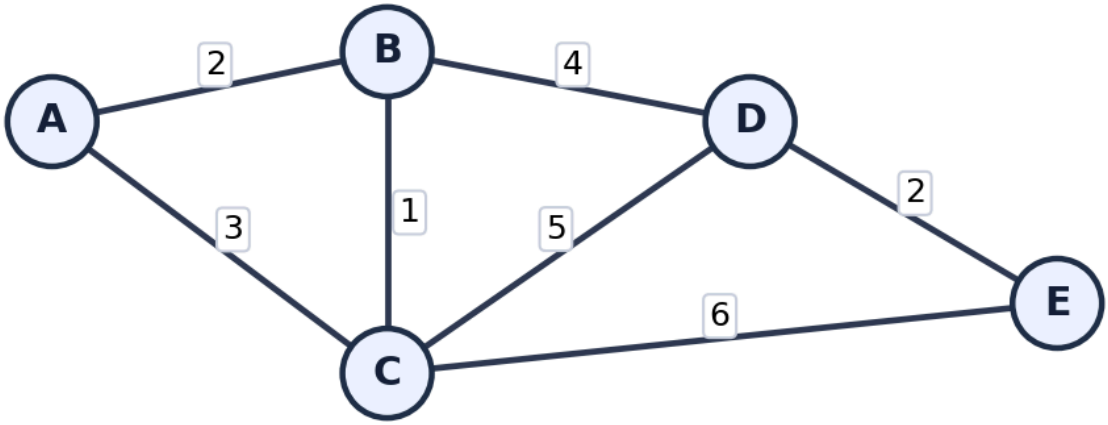
(C) `S, B, C, A, D`

(D) `S, A, C, B, D`

(B)

The final distances become S=0, B=1, A=3, C=4, D=7, so vertices are finalized in that order.

(10) Prim's algorithm starts from `A` on the following undirected weighted graph.



Which sequence of edges is selected?

(A) `(A, C), (C, B), (B, D), (D, E)`

(B) `(A, B), (B, C), (B, D), (D, E)`

(C) `(A, B), (A, C), (C, D), (D, E)`

(D) `(A, C), (C, D), (D, E), (B, C)`

(B)

The selected edges are A-B (2), B-C (1), B-D (4), and D-E (2), with total cost 9.

(11) A min-heap is stored in a 1-indexed array:

```
index:  1  2  3  4  5  6
value:  3  8  5 14 10  9
```

Which statement is TRUE?

(A) The parent of the value `10` is `8`.

(B) The parent of the value `10` is `5`.

(C) The value `14` is the right child of `5`.

(D) The array violates the min-heap property because `8 > 5`.

(A)

Value 10 is at index 5, whose parent is at index floor(5/2)=2, the value 8.

(12) Which traversal of a binary search tree visits the keys in ascending sorted order?

(A) Preorder traversal

(B) Inorder traversal

(C) Postorder traversal

(D) Level-order traversal

(B)

For a binary search tree, inorder traversal visits left subtree, root, then right subtree, producing sorted keys.

B. Short-answer questions, please provide the derivation for each question along with your answer (8% each, 40%).

(13) Briefly explain each of the following terms.

(a) Base case

(b) Stable sort

(c) Adjacency list

(d) Complete binary tree

(a) A **base case** is a condition that stops a recursive function from calling itself again.

(b) A **stable sort** preserves the relative order of records with equal keys.

(c) An **adjacency list** represents a graph by storing, for each vertex, the vertices adjacent to it, often with edge weights when needed.

(d) A **complete binary tree** is a binary tree in which every level is full except possibly the last, and the last level is filled from left to right.

(14) Suppose a hash table has size 7 and uses separate chaining. The hash function is `h(k) = k % 7`, and new keys are appended to the end of each chain.

Insert the keys in this order:

```
10, 21, 32, 14, 25, 17, 6
```

Show the final contents of every bucket from index 0 to index 6.

The hash values are:

```
10 % 7 = 3
21 % 7 = 0
32 % 7 = 4
14 % 7 = 0
25 % 7 = 4
17 % 7 = 3
6  % 7 = 6
```

Final table:

```
0: [21, 14]
1: []
2: []
3: [10, 17]
4: [32, 25]
5: []
6: [6]
```

(15) Insert the following keys into an initially empty min-heap:

`13, 6, 18, 2, 9, 20, 4`

Show the array representation of the heap after all insertions. Include the important percolation steps.

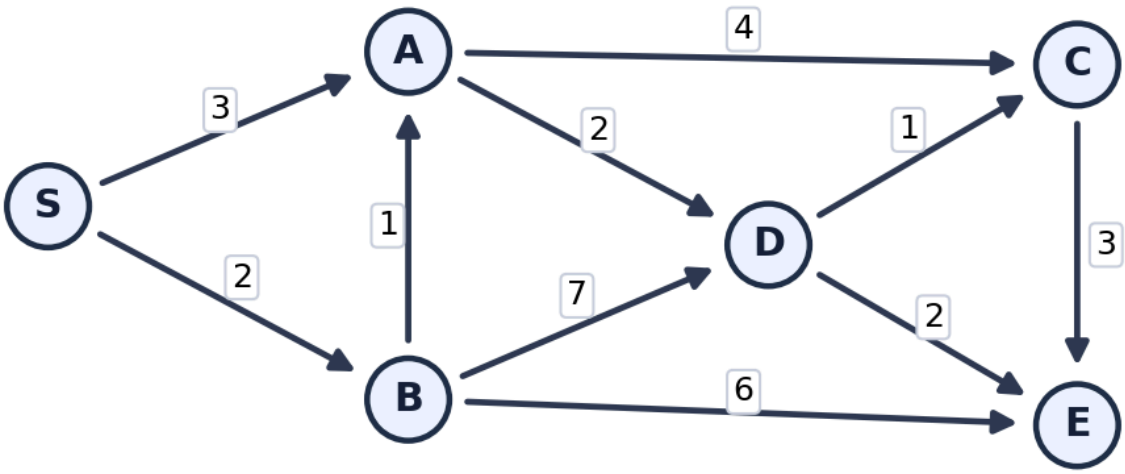
Insertion trace:

```
13 -> [13]
6  -> [6, 13]
18 -> [6, 13, 18]
2  -> [2, 6, 18, 13]
9  -> [2, 6, 18, 13, 9]
20 -> [2, 6, 18, 13, 9, 20]
4  -> [2, 6, 4, 13, 9, 20, 18]
```

Final min-heap array:

`[2, 6, 4, 13, 9, 20, 18]`

(16) Run Dijkstra's algorithm from `S` on the following directed weighted graph.



List the vertices in the order they are finalized and give the final shortest-path distances from `S`.

The finalized order is:

`S, B, A, D, C, E`

Final distances:

```
S: 0
B: 2
A: 3
```

```
D: 5
C: 6
E: 7
```
The key updates are: from S get A=3 and B=2; from B keep A=3 and set D=9, E=8; from A improve D to 5 and set C=7; from D improve C to 6 and E to 7.

(17) A binary tree has the following traversal sequences:

```
Preorder: M F B H T R Z
Inorder:  B F H M R T Z
```
Can the tree be constructed uniquely? If yes, draw the tree. If not, explain why not.

Yes. Preorder gives the root first, so `M` is the root. Inorder splits the left subtree as `B F H` and the right subtree as `R T Z`.

The left preorder part is `F B H`, so `F` is the left child of `M`, with `B` as its left child and `H` as its right child. The right preorder part is `T R Z`, so `T` is the right child of `M`, with `R` as its left child and `Z` as its right child.