Option in red is the correct answer

Q1. Which of the following functions is suitable as a nonlinear activation function for training neural networks?

A. $g(x) = \min(x, 6)$ (a shifted negated ReLU)

B. $g(x) = 2x + 1$ (this is a linear function)

C. $g(x) = 1$ if $x > 0$ and -1 otherwise (the gradient is almost always 0, not good for training)

D. $g(x) = x$ (a linear function)

Q2. Which of the following is TRUE regarding the nonlinear activation functions (such as Sigmoid and ReLU) within neural networks?

A. They can speed up the gradient calculation in backpropagation, as compared to linear units. (Not true, they are typically more expensive)

B. They help to learn nonlinear decision boundaries. (correct. Without them the decision boundary is linear and cannot solve many nonlinear decision problems like XOR.)

C. They must always output positive values. (Not true. For example, leaky ReLU)

D. They are applied only to the hidden units. (e.g., Sigmoid can be on the output unit)

Q3. If the neural network output is $y = (x^2 + w_1)^2 \cdot w_2$. What's the derivative/gradient for w_1?

A. 1

B. $x^2 + 1$

C. $w_2 \cdot (x^2 + 1)$

D. $w_2 \cdot 2(x^2 + w_1)$ (view the function as a single variable function in w_1, and view the other variable as constants)

Q4. What could be a possible scenario if we are running a breadth-first search (BFS) on an infinite tree-structured state-space graph? Suppose we do NOT have assumptions on the existence of the goal states or on the edge costs. Multiple answers are possible.

A. Gets stuck in an infinite loop. (Yes, when there is no goal state in the tree)

B. Visits a finite number of nodes, then return one at random. (No, the algorithm doesn't stop before exhausting all nodes or find a goal state. It doesn't return one at random)

C. Finds the least-cost goal. (correct. For example, when there is a goal state finite step away from the initial state, and the edge cost is all 1.)

D. Finds a goal, but not necessarily the least cost goal. (correct, since we don't have assumption on the edge cost.)

Q5. What could be a possible scenario if we are running a uniform-cost search (UCS) on an infinite tree-structured state-space graph? Suppose we do NOT have assumptions on the existence of the goal state, but we assume that the edge cost is always $\geq \epsilon > 0$. Multiple answers are possible.

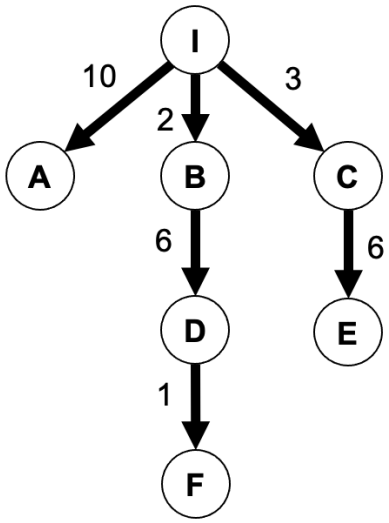
A. Gets stuck in an infinite loop. (Yes, when there is no goal state in the tree)

B. Visits a finite number of nodes, then return one at random. (No, the algorithm doesn't stop before exhausting all nodes or find a goal state. It doesn't return one at random)

C. Finds the least-cost goal. (correct. For example, when there is a goal state finite step away from the initial state, and the edge cost is all 1.)

D. Finds a goal, but not necessarily the least cost goal. (No. When there is a goal state finite step away from the initialization, UCS finds the least-cost goal. Otherwise, the algorithm doesn't stop.)

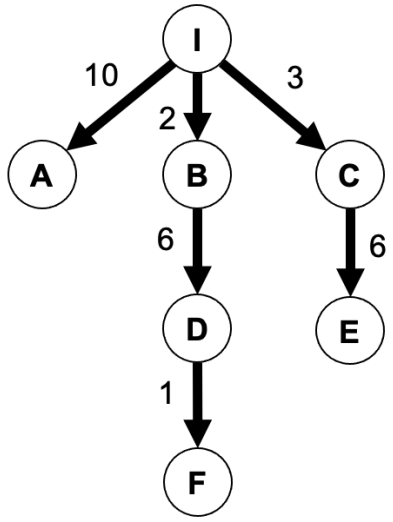
Q6. Consider the following search tree. Initial state is I, and the goal state is D. Suppose we run BFS. Write down for each iteration, the node expanded, and the fringe at the end of the iteration. The fringe should be sorted, with the front on the left-hand side. For tie-breaking, following the dictionary order (e.g., A before B).



Answer:
Iteration: expanded node, fringe
1: I, [A, B, C]

2: A, [B, C]
3: B, [C, D]
4: C, [D, E]
5: D, [E]

Q7. Consider the following search tree. Initial state is I, and the goal state is D. Suppose we run UCS. Write down for each iteration, the node expanded, and the fringe at the end of the iteration. The fringe need not be sorted. The expanded node, and the nodes in the fringe should be in the format of (node id, cost). For tie-breaking, following the dictionary order (e.g., A before B).



Answer:
Iteration: expanded node, fringe
1: (I,0), [(A,10), (B,2), (C,3)]
2: (B,2), [(A,10), (C,3), (D,8)]
3: (C,3), [(A,10), (D,8), (E,9)]
4: (D,8), [(A,10), (E,9)]