Q1. Given two admissible heuristic functions h1 and h2, h2 dominates h1 if for any state s, h1(s) <= h2(s) <= h*(s), where h*(s) is the true cost to the goal from a state s. For use in A* search, why do we prefer the dominating heuristic function h2 over h1?

A. We don't. In A* we always want to be optimistic, so a dominating heuristic is less optimistic and so may not work better.

B. A dominating heuristic is easier to compute since it is closer to the true heuristic.

C. A dominating heuristic is closer to the true heuristic and thus gives more accurate estimation of the costs.

D. A dominating heuristic must have a simpler form than the other heuristic.

Q2. What is the main difference between Uninformed Search (e.g., uniform cost search) and Informed Search (e.g., A* search)?

A. Informed Search does not know the edge costs but has a guess for path costs called a heuristic.

B. Uninformed Search does not know the edge costs but Informed does.

C. Informed Search has the same information as Uniformed Search but also a guess for the distance to the goal called a heuristic.

D. Informed Search has access to a probability distribution to inform most likely paths

Q3. Which of the following algorithm would be most likely to get stuck in a poor local optimum?

A. Hill climbing without random restart

B. Hill climbing with random restart

C. Simulated annealing

D. Genetic algorithm

Q4. Which of the following mechanism in genetic algorithm allows it to favor states with better fitness value?

A. Cross-over

B. Mutation

C. Natural selection

D. Random restart

Q5. Consider a value function f on a finite state space. f only has one local optimum that is also the global optimum. And for any states s1 and s2, f(s1) is not equal to f(s2). When starting from the same initial state, which of the following algorithms will always reach the global optimum of f? Multiple answers are possible.

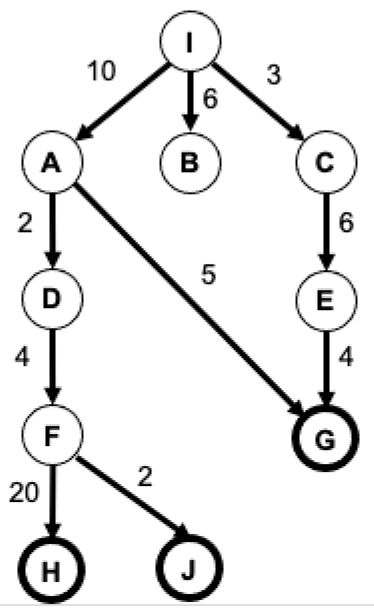
A. Hill climbing without random restart.

B. Hill climbing with random restart.

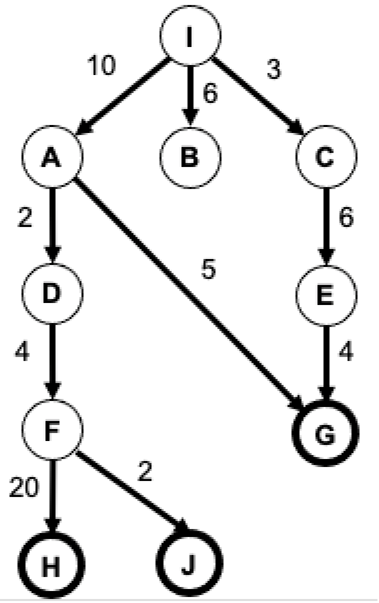
C. Simulated Annealing.

D. Genetic algorithm.

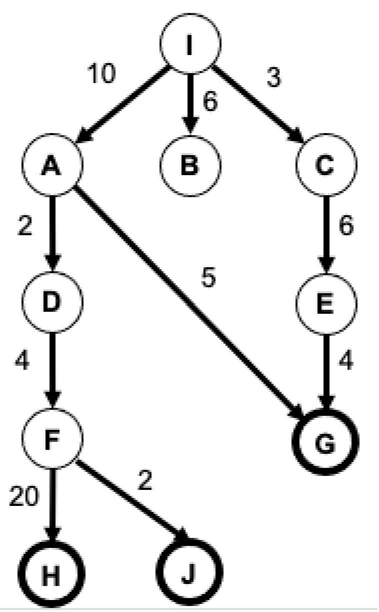
Q6. Consider the state space graph below. The initial state is I, and goal states have bold borders. Nodes are expanded left to right when there are ties. Suppose we run BFS. Write down for each iteration, the node expanded, and the fringe at the end of the iteration.



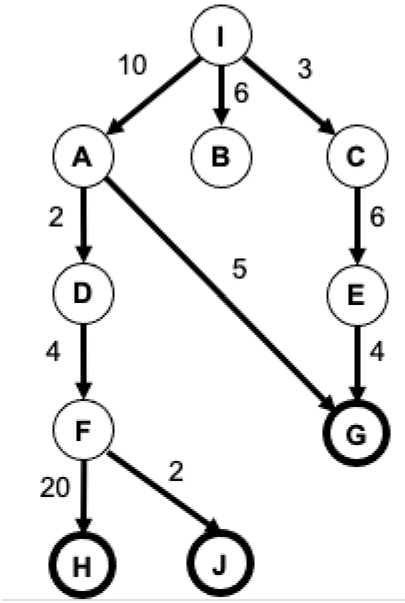
Q7. Consider the state space graph below. The initial state is I, and goal states have bold borders. Nodes are expanded left to right when there are ties. Suppose we run DFS. Write down for each iteration, the node expanded, and the fringe at the end of the iteration.



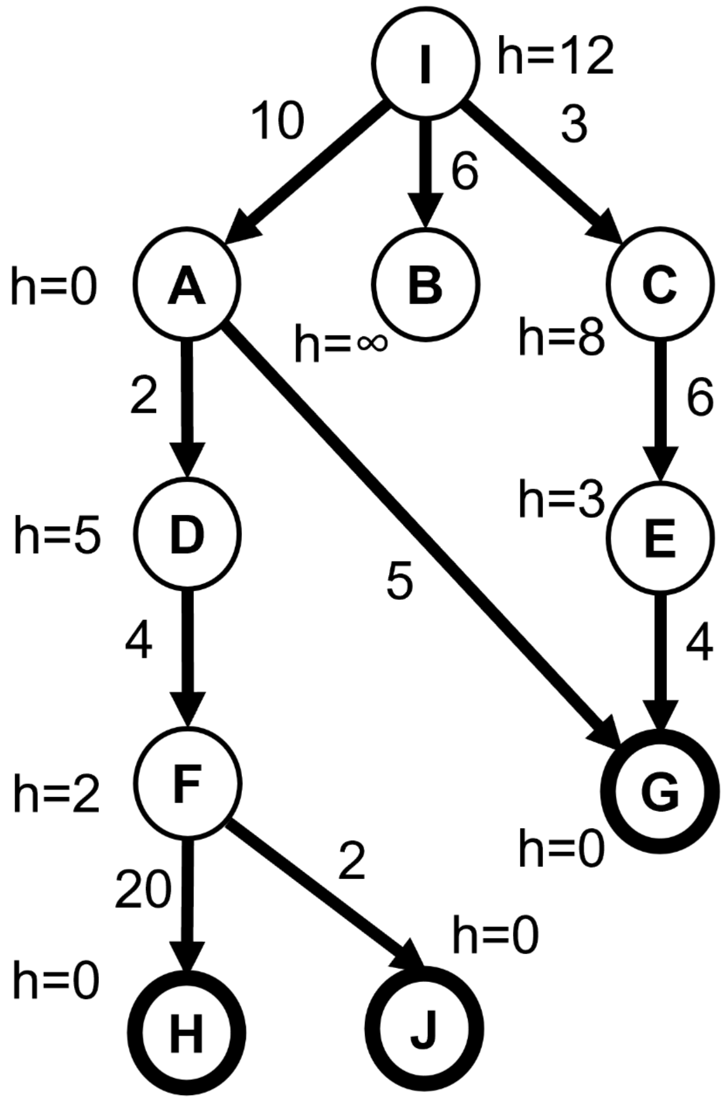
Q8. Consider the state space graph below. The initial state is I, and goal states have bold borders. Nodes are expanded left to right when there are ties. Suppose we run UCS. Write down for each iteration, the node expanded, and the fringe at the end of the iteration. The fringe need not be sorted. The expanded node, and the nodes in the fringe should be in the format of (node id, cost). When there are multiple copies of a node encountered, keep the copy with the smaller cost.



Q9. Consider the state space graph below. The initial state is I, and goal states have bold borders. Nodes are expanded left to right when there are ties. Suppose we run IDS (Iterative Deepening Search). Write down for each iteration in each stage, the node expanded, and the fringe at the end of the iteration.



Q10. Consider the state space graph below, with heuristic h next to each state. The initial state is I, and goal states have bold borders. Nodes are expanded left to right when there are ties. Suppose we run A*. Write down for each iteration in each stage, 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, g+h value). When there are multiple copies of a node encountered, keep the copy with the smaller cost.



Q11. Consider a state space with only five states and their corresponding scores (1, 3), (2, 5), (3, 2), (4, 3), (5, 1), for which the first element is the state and the second element is the score. Thus, (1, 3) indicates state 1 with a score 3. Each state i has two neighbors (i+1) and (i-1), except state 1 has only one neighbor state 2, and state 5 only has one neighbor state 4. Consider running hill-climbing with random restart to find the state with the maximum score. If we always start or restart from a state sampled from all states with equal probability, what is the probability of finding the best state in two runs?