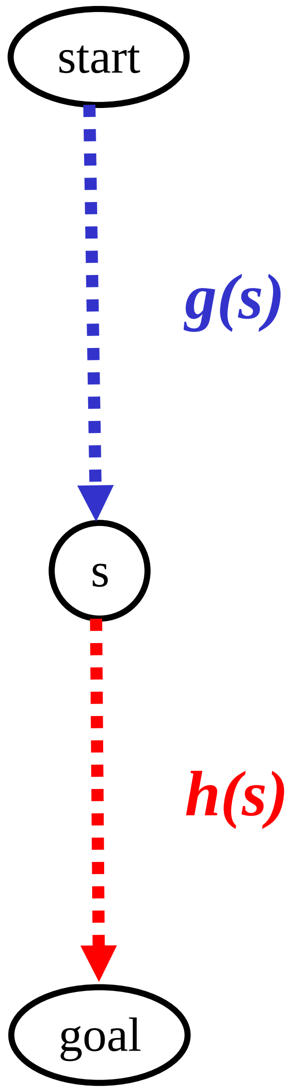
Q1-1: Your search algorithm uses a priority queue to track the state with the min score to pop next. If $h(s)$ is an <u>admissible heuristic</u>, what score should you use in the priority queue? Rank them from best to worst.

1. $g(s)$+h(s), $h(s)$, $g(s)$

2. $g(s)$+h(s), $g(s)$, $h(s)$

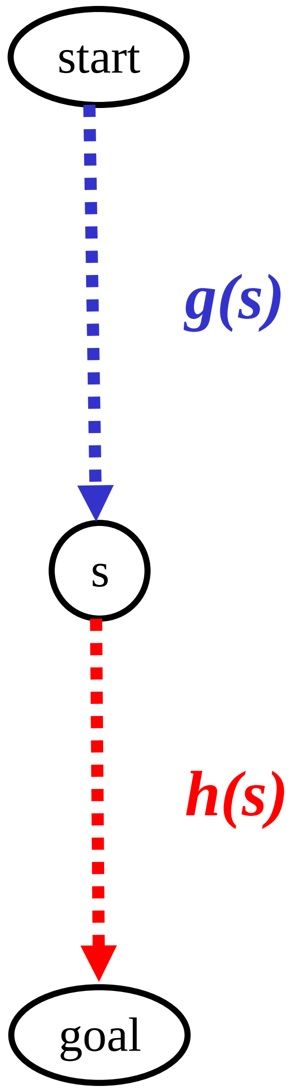
3. $g(s)$, $h(s)$, $g(s)$+h(s)

4. $g(s)$, $g(s)$+h(s), $h(s)$



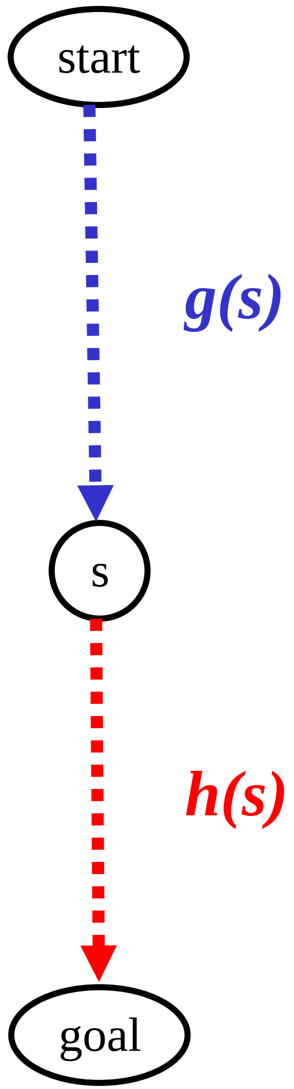
Q1-1: Your search algorithm uses a priority queue to track the state with the min score to pop next. If *h*(*s*) is an <u>admissible heuristic</u>, what score should you use in the priority queue? Rank them from best to worst.

1. *g*(*s*)+h(*s*), *h*(*s*), *g*(*s*)
2. *g*(*s*)+h(*s*), *g*(*s*), *h*(*s*)
3. *g*(*s*), *h*(*s*), *g*(*s*)+h(*s*)
4. *g*(*s*), *g*(*s*)+h(*s*), *h*(*s*)



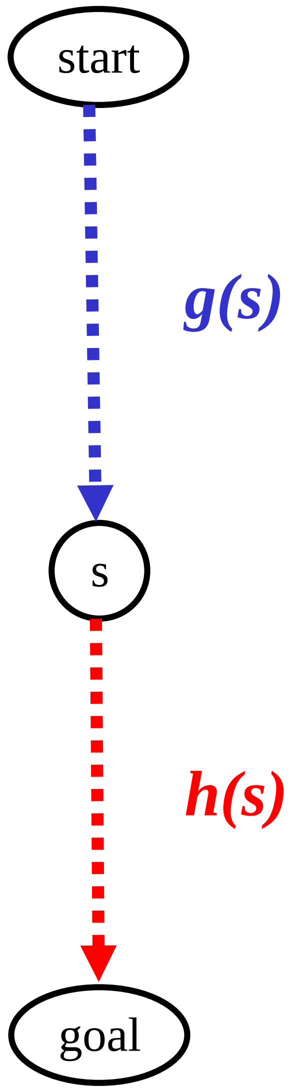
Q1-2: Your search algorithm uses a priority queue to track the state with the min score to pop next. If $h(s)$ is a heuristic (not necessarily admissible), what score should you use in the priority queue? Rank them from best to worst.

1. $g(s)$+h(s), $h(s)$, $g(s)$
2. $g(s)$+h(s), $g(s)$, $h(s)$
3. $g(s)$, $h(s)$, $g(s)$+h(s)
4. $g(s)$, $g(s)$+h(s), $h(s)$



Q1-2: Your search algorithm uses a priority queue to track the state with the min score to pop next. If $h(s)$ is a heuristic (<u>not necessarily admissible</u>), what score should you use in the priority queue? Rank them from best to worst.

1. $g(s)$+h(s), $h(s)$, $g(s)$
2. $g(s)$+h(s), $g(s)$, $h(s)$
3. $g(s)$, $h(s)$, $g(s)$+h(s)
4. $g(s)$, $g(s)$+h(s), $h(s)$ ⬅



Q1-3: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let $h(s)$ be the number of hours needed to ride a bike from city s to your destination. $h(s)$ is

1. Not a valid heuristic

2. A valid heuristic but not admissible

3. An admissible heuristic

Q1-3: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let $h(s)$ be the number of hours needed to ride a bike from city s to your destination. $h(s)$ is

1. Not a valid heuristic

2. A valid heuristic but not admissible

3. An admissible heuristic

Q1-4: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let $h(s)$ be the current temperature in city s. $h(s)$ is

1. Not a valid heuristic
2. A valid heuristic but not admissible
3. An admissible heuristic

Q1-4: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let *h*(*s*) be the current temperature in city *s*. *h*(*s*) is

1. Not a valid heuristic
2. A valid heuristic but not admissible
3. An admissible heuristic

Q1-5: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let $h(s)$ be the number of hours to drive from s to your destination if you ignore speed limits. $h(s)$ is

1. Not a valid heuristic
2. A valid heuristic but not admissible
3. An admissible heuristic

Q1-5: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let $h(s)$ be the number of hours to drive from s to your destination if you ignore speed limits. $h(s)$ is

1. Not a valid heuristic
2. A valid heuristic but not admissible
3. An admissible heuristic ←

Q2-1: Consider two heuristics for the 8 puzzle problem. *h1* is the number of tiles in wrong position. *h2* is Manhattan distance between tile and its goal location. How do *h1* and *h2* relate?

1. *h1* dominates *h2*

2. *h2* dominates *h1*

3. Neither heuristic dominates the other

Q2-1: Consider two heuristics for the 8 puzzle problem. *h1* is the number of tiles in wrong position. *h2* is Manhattan distance between tile and its goal location. How do *h1* and *h2* relate?

1. *h1* dominates *h2*

2. *h2* dominates *h1*

3. Neither heuristic dominates the other

Q2-2: Consider finding the fastest driving route from one US city to another, as before. Let $h(s)$ be the minimum number of hours to drive from s to your destination, returned from 1000s of physics simulations of driving in different weather conditions. $h(s)$ is

1. Not a valid heuristic

2. A valid heuristic but not admissible

3. An admissible heuristic

4. An admissible heuristic but a poor choice for A*

Q2-2: Consider finding the fastest driving route from one US city to another, as before. Let *h*(*s*) be the minimum number of hours to drive from *s* to your destination, returned from 1000s of physics simulations of driving in different weather conditions. *h*(*s*) is

1. Not a valid heuristic

2. A valid heuristic but not admissible

3. An admissible heuristic

4. An admissible heuristic but a poor choice for A*

Q2-3: A* search places expanded states in the CLOSED data structure. What is the space complexity of CLOSED?

1. O(1)
2. O(depth of optimal goal)
3. O(log number of states)
4. O(number of states)

Q2-3: A* search places expanded states in the CLOSED data structure. What is the space complexity of CLOSED?

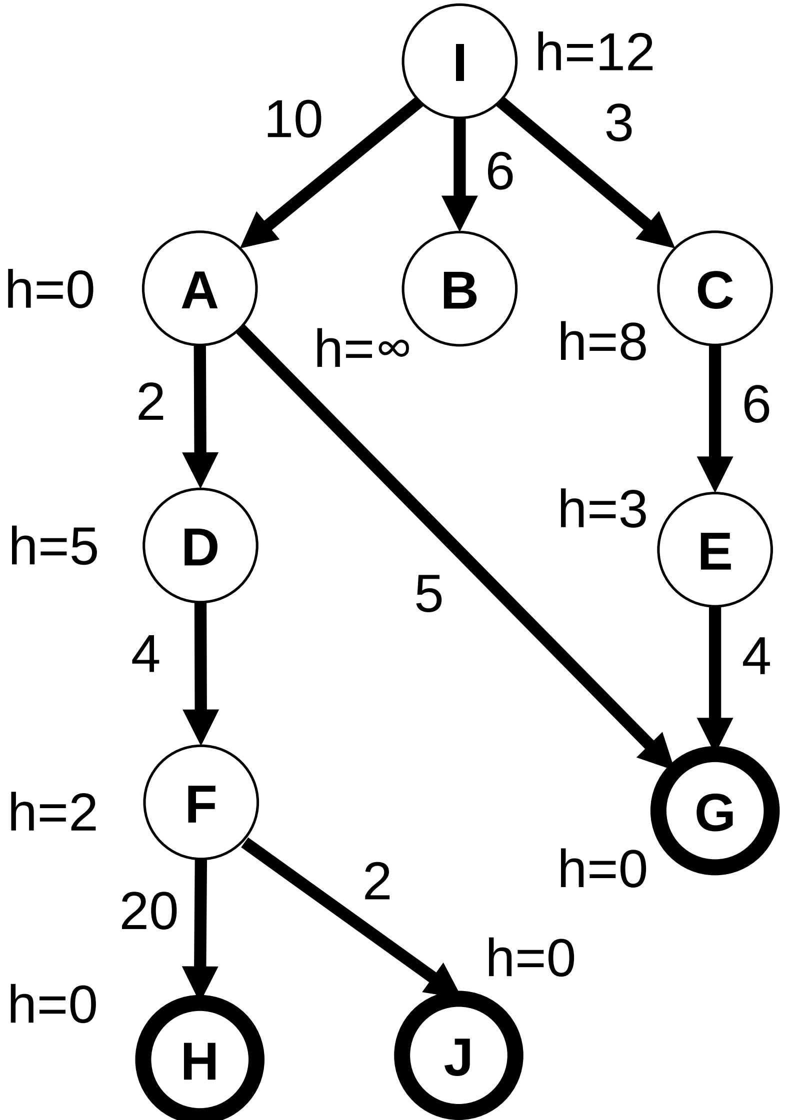
1. O(1)

2. O(depth of optimal goal)

3. O(log number of states)

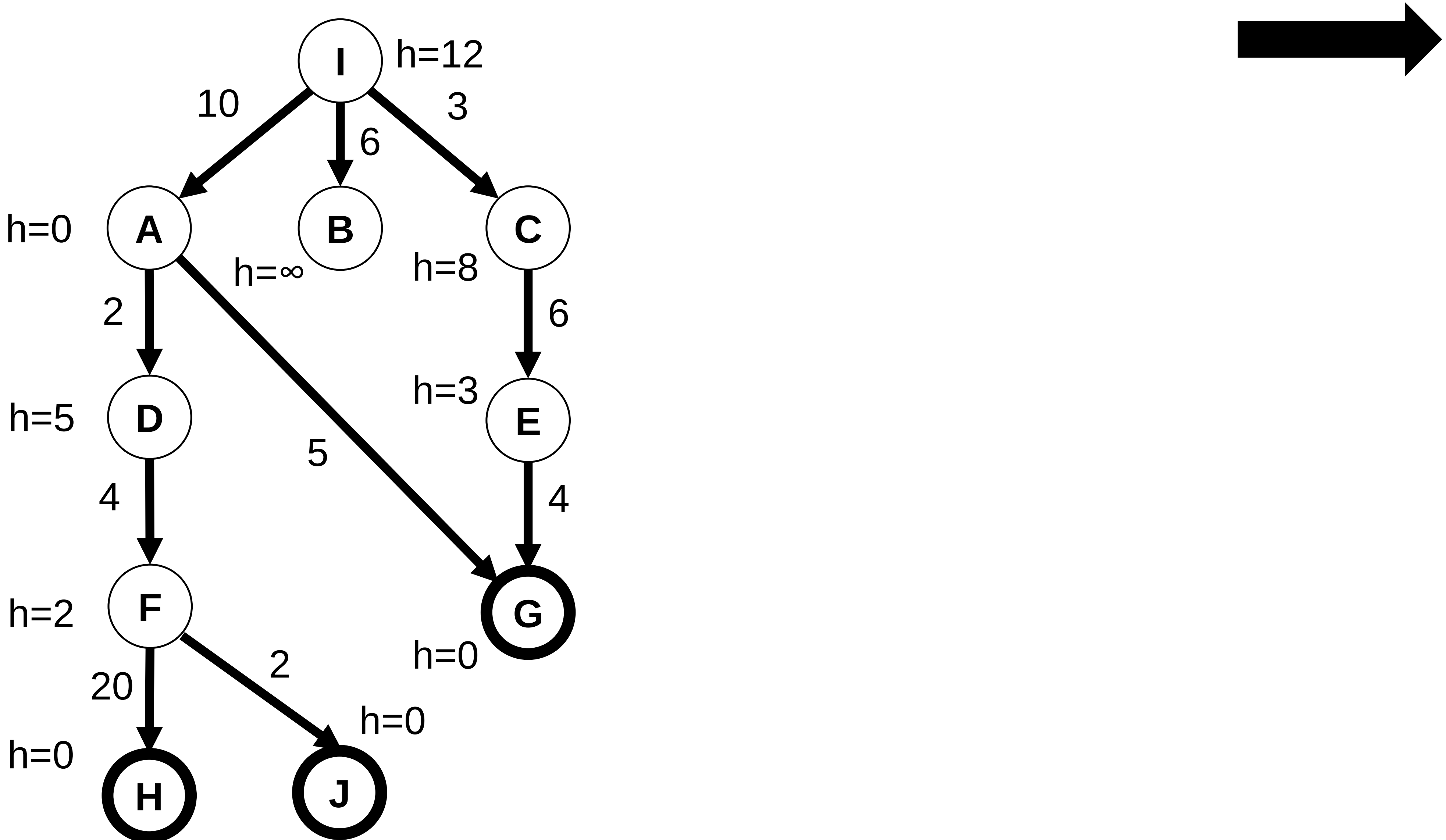
4. O(number of states)

Q2-4: Consider the state space graph below. Goal states have **bold** borders. *h*(*s*) is show next to each node. What node will be expanded by A* after the initial state **I**?



1. A
2. B
3. C

Q2-4: Consider the state space graph below. Goal states have **bold** borders. *h*(*s*) is show next to each node. What node will be expanded by A* after the initial state **I**?



1. A
2. B
3. C

Q3-1: How does IDA* save space compared to A*?

1. IDA* only tracks the best *k* nodes in the priority queue

2. IDA* will not get stuck exploring deep, suboptimal paths

3. IDA* uses a better heuristic and expands fewer nodes

Q3-1: How does IDA* save space compared to A*?

1. IDA* only tracks the best *k* nodes in the priority queue

2. IDA* will not get stuck exploring deep, suboptimal paths

3. IDA* uses a better heuristic and expands fewer nodes

Q3-2: You are running IDA* in a state space graph with positive real-valued *f*(*s*) scores increasing threshold *t* as described. When *t* increases and IDA* restarts, now many new nodes are expanded?

1. 0 or 1 new nodes

2. Exactly 1 new node

3. 1 or more new nodes

Q3-2: You are running IDA* in a state space graph with positive real-valued *f*(*s*) scores increasing threshold *t* as described. When *t* increases and IDA* restarts, now many new nodes are expanded?

1. 0 or 1 new nodes

2. Exactly 1 new node

3. 1 or more new nodes ⬅