Tick those statements that you believe to be true:

☐ A heuristic function *h* maps a state to a single number, namely, the estimated distance to the goal.

☐ For every goal state *s*, the value of the heuristic function is 1 ("true"), that is, *h*(*s*) = 1.

☐ In best-first search, the heuristic function *h* is always used as the evaluation function *f*, that is, to determine which node to expand next.

☐ Best-first search always expands the fringe node with the highest *f*-value next.

☐ Greedy best-first search always returns an optimal solution, that is, the shortest path from the initial state to a goal state.

Submit

Skip

Greedy Best-First Tree Search

Apply the greedy best-first search algorithm to the Touring Romania problem, this time starting from Oradea. To do this, you will need the map and the heuristic function defined for this problem. Apply the tree-search version of the algorithm, that is, treat repeated occurrences of the same state as new search nodes.

To generate the answer, make a string out of the first letter of each city name in the order in which the nodes are expanded.

The string will start with O for Oradea. The next letter will be S or Z, depending on which node the algorithm will expand next. The last letter in the string should be B for Bucharest. Note, letters should be added for nodes that are expanded, not for each successor that is generated.

Type your answer in the box below. Don't leave any spaces between the letters.

Submit

Skip

Tick those statements that you believe to be true:

- ☐ The value of *f*(*s*) for a state *s* in the A* algorithm represents the cost of the cheapest path from the initial state through state *s* to a goal state.
- ☐ A* is optimal given an admissible heuristic. This means no other algorithm generates a smaller search tree.
- ☐ A* is optimal given an admissible heuristic. This means no other algorithm can find a shorter path to a goal state.
- ☐ A* is optimal given an admissible heuristic. This means no other algorithm can find a goal state faster.

Submit

Skip

A* Tree Search

Apply the A* search algorithm to the Touring Romania problem, this time starting from Oradea. To do this, you will need the map and the heuristic function defined for this problem. Apply the tree-search version of the algorithm, that is, treat repeated occurrences of the same state as new search nodes.

To generate the answer, make a string out of the first letter of each city name in the order in which the nodes are expanded.

The string will start with O for Oradea. The next letter will be S or Z, depending on which node the algorithm will expand next. The last letter in the string should be B for Bucharest. Note, letters should be added for nodes that are expanded, not for each successor that is generated. Don't leave any spaces between the letters.

Type your answer in the box below. Don't leave any spaces between the letters.

Submit

Skip

Tick those statements that you believe to be true:

☐ A heuristic $h(n)$ is admissible if it never overestimates the distance from n to the nearest goal node.

☐ The heuristic function that is $h(n) = 0$ for all nodes n is admissible.

☐ A* using tree search is optimal if the heuristic $h(n)$ is admissible.

☐ The more accurate the heuristic $h(n)$ given to A* is, the better the solutions the algorithm will find will be.

Submit

Skip

Tick those statements that you believe to be true:

- ☐ Breadth-first search (uninformed search with FIFO strategy) needs exponential memory in the worst case.
- ☐ Depth-first search (uninformed search with LIFO strategy) needs exponential memory in the worst case.
- ☐ With an admissible heuristic, A* graph search needs exponential memory in the worst case.
- ☐ With a perfect heuristic, A* graph search needs exponential memory in the worst case.
- ☐ If the search space is a two-dimensional grid, A* graph search needs exponential memory in the worst case.

Submit

Skip

Given the definitions of the predicates (relations) for our DWR domain you have just seen, which of the following assertions are consistent with these definitions?

- ☐ `(belong crane1 loc1)`
- ☐ `(at cont1 loc1)`
- ☐ `(occupied loc1 robot1)`
- ☐ `(holding crane1 cont1)`
- ☐ `(on cont1 cont1)`

Submit

Skip

Take a look at the initial state defined in a simple DWR problem. You can ignore the goal. Tick those statements that you believe to be true:

- ☐ `(on cb pallet)` holds in the initial state.
- ☐ `(in ce q1)` holds in the initial state.
- ☐ The initial state satisfies {`(in ca p1)`, `(not (in cb q1))`}.
- ☐ initial state |= {`(in ca p1)`, `(not (in cd q1))`}

Submit

Skip

Define the following two operators (described earlier) in the PDDL syntax.

- crane *k* at location *l* loads container *c* onto robot *r*
- crane *k* at location *l* puts container *c* onto *d* in pile *p*

Continue

Tick those statements that you believe to be true:

- ☐ An action *a* is applicable in a state *s* if and only if *s* satisfies the preconditions of *a*.
- ☐ The `move` operator can be instantiated to an action that has the same atom as a positive and negative effect.
- ☐ If *a* is applicable in *s* and *a* has effects (*e*) and (not *e*) then *e* holds in $\gamma(s, a)$.
- ☐ A relation that does not appear as an effect in any operator is superfluous and can be deleted.

Submit

Skip

For this question, you need look at a random planning domain and problem. Which of the following actions are applicable in the initial state?

- ☐ `( op1 B B C )`
- ☐ `( op1 A A A )`
- ☐ `( op1 B A B )`
- ☐ `( op2 C B A )`
- ☐ `( op2 C B B )`

Submit

Skip

Lecture 2.8c State Space Search - Domains and Problems [03:34]

And here is a question that requires some knowledge of combinatorics. How many minimal solutions are there for the DWR problem with 6 containers and 1 robot described in PDDL earlier?

If you feel this is too hard, just skip the quiz. The idea behind this quiz is simply to familiarize you more with the DWR domain. And remember, in-video quizzes are there to help you and do not count towards your final result.

Submit

Skip

Tick those statements that you believe to be true:

- ☐ For a STRIPS planning problem (Σ, s_i, g) with $\Sigma = (S, A, \gamma)$, the following holds: $\Gamma^{>}(s_i) = S$.
- ☐ There exists a k such that $\Gamma^{>}(s_i) = \Gamma^{k}(\{s_i\})$.
- ☐ $\Gamma^{>}(\varnothing) = s_i$
- ☐ A planing problem (Σ, s_i, g) has a solution if and only if there exists a k such that $S_g \cap \Gamma^{k}(\{s_i\}) \neq \varnothing$.

Submit

Skip

For this question, you need look at a random planning domain and problem. Which of the following actions are relevant for the goal given in the problem?

☐ `( op1 A A A )`

☐ `( op1 A A C )`

☐ `( op1 A B A )`

☐ `( op2 B B C )`

☐ `( op2 A B A )`

Submit

Skip

Try to modify the pseudo code for the forward search planning algorithm to turn it into a backward search algorithm.

Continue