Solutions to Mid-term Sample Questions

Problem 1

Can an environment be both known and unobservable? Give an example.

Solution

An environment can be both known and unobservable. An example of this is Russian roulette, a potentially lethal game of chance in which a player places a single round in a revolver, spins the cylinder, places the muzzle against his head, and pulls the trigger.

The environment is known beforehand (i.e., each agent knows what actions are possible, what states of the world are possible, and what effect will follow from each possible combination of a world state + action), but the revolver state is unobservable from agents (they cannot observe where the bullet is located).

Another example could be the maze environment from MP1. The agent may be given the definition of the environment (the maze) but have no sensing capabilities. Such agent could still find the solution to the maze and execute the solution, assuming the agent can execute perfectly.

Problem 2

What is the distinction between a world state and a search tree node?

Solution

A world state is a description or "snapshot" of the world. A search tree node is part of the search tree data structure. It contains a world state along with other information (heuristic function value, evaluation function value, parent pointer, etc.). In general, it is possible for multiple tree nodes to contain the same world state.

Problem 3

In the tree search formulation, why do we restrict step costs to be non-negative?

Solution

If there is a loop with a net negative cost in the state space, a search algorithm can keep going around this loop infinitely and lowering its cost every time.

Problem 4

For each type of maze described below, specify whether breadth-first-search (BFS) or depth-first-search (DFS) will more efficiently find a solution in the worst case, and say why. Assume that both BFS and DFS return the first solution path they find.

a. The Albuquerque maze has 3 possible directions that you can take at each intersection. No path is longer than 25 steps. There is only one solution.

Solution

If you assume that the correct solution has a depth of 25, then DFS and BFS have exactly the same worst-case time complexity, about 3^25. If you assume that the correct solution might have a depth d<25, then BFS is more efficient in the worst case.

b. The Belmont maze has 3 possible directions that you can take at each intersection. No path is longer than 25 steps. About half of all available paths are considered solutions to the maze.

Solution

In the worst case, DFS will explore about half the paths in the tree, while BFS will explore all the paths up to the depth of the optimal solution, d. It is difficult to say which one will be more efficient without knowing d.

c. The Crazytown maze has 3 possible directions that you can take at each intersection. The maze is infinite in size, so some paths have infinite length. There is only one correct solution, which is known to require 25 steps.

Solution

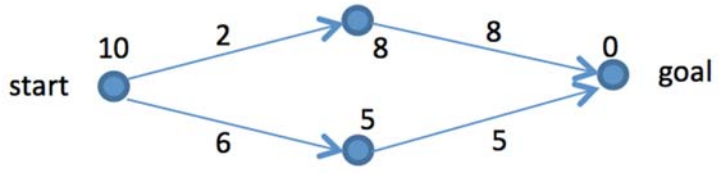
BFS is better, because its complexity is only about 3^25. DFS has a complexity of O{b^m} where m is much greater than 25 and could even be infinite (assuming no repeated state detection).

Problem 5

Suppose you are given a "perfect" heuristic function that gives the correct optimal distance from each node to the goal. Is greedy best-first search with this heuristic optimal? If not, give a counterexample.

Solution

It is not optimal. Counterexample is below.



(nodes are marked with heuristic function value, edges are marked with step costs)

Problem 6

Explain why it is a good heuristic to choose the variable that is *most* constrained but the value that is *least* constraining in a CSP search.

Solution

By choosing the variable that is most constrained we are minimizing the branching factor of backtracking search. We are also selecting the variable that is most likely to cause a failure soon. This helps us prune our search tree by avoiding pointless searches through other variables when one fails. By choosing the value that is least constraining we are maximizing the possible options for neighboring variables and giving our search more flexibility and maximizing the possibility of finding a solution.

Problem 7

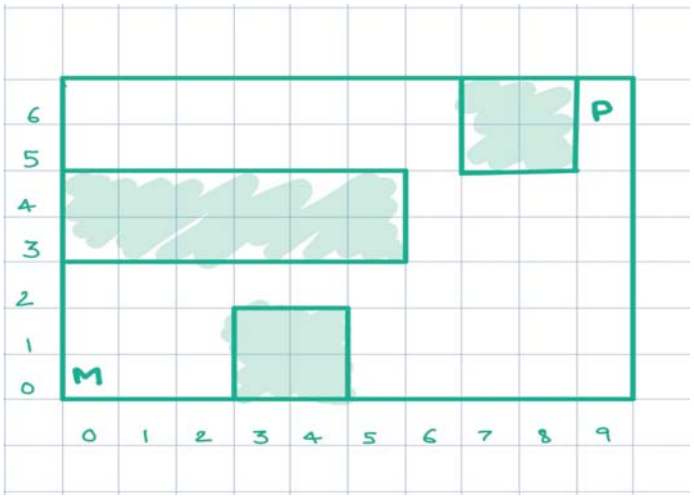
What is local search for CSPs? For which kinds of CSPs might local search be better than backtracking search? What about the other way around?

Solution

Local search for CSPs is to assign a value to every variable at the initial state, and the search changes the value of one variable at a time. Local search (i.e., hill climbing) may be a good choice when the problem is relatively loosely constrained and there are many possible solutions. An example of such a problem is n-queens. Backtracking search may be better for more tightly constrained problems with few possible solutions, such as sudoku. In such problems, it is also difficult to come up with local modifications that can remove constraint violations.

Problem 8

Refer to the maze shown below. Here, 'M' represents Mario, 'P' represents Peach, and the goal of the game is to get Mario and Peach to find each other. In each move, *both* Mario and Peach take turns. For example, *one* move would consist of Peach moving a block to the bottom from her current position, and Mario moving one block to the left from his current position. Standing still is also an option.



a. Describe state and action representations for this problem.
b. What is the branching factor of the search tree?
c. What is the size of the state space?
d. Describe an admissible heuristic for this problem.

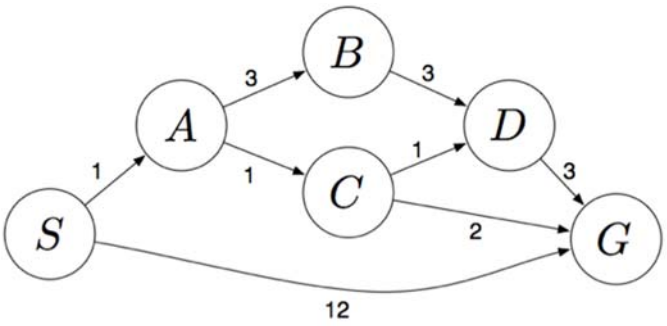
Solution

a. State = [mx,my,px,py] where (mx,my)=position of Mario, (px,py)= position of Peach. Action ϵ {ml,mr,mu,md,ms}X{pl,pr,pu,pd,ps} where the first letter denotes the person moving (Mario or Peach), the second letter denotes the direction of movement (left, right, up, down, or stay).

b. 25
c. (54-20)x(54-20-1)=34x33=1122
d. Manhattan distance: |mx-px|+|my-py|

Problem 9

Consider the search problem with the following state space:



S denotes the start state, G denotes the goal state, and step costs are written next to each arc. Assume that ties are broken alphabetically (i.e., if there are two states with equal priority on the frontier, the state that comes first alphabetically should be visited first).

a. What path would BFS return for this problem?
b. What path would DFS return for this problem?
c. What path would UCS return for this problem?
d. Consider the heuristics for this problem shown in the table below.

State	h_1	h_2
S	5	4
A	3	2
B	6	6
C	2	1
D	3	3
G	0	0

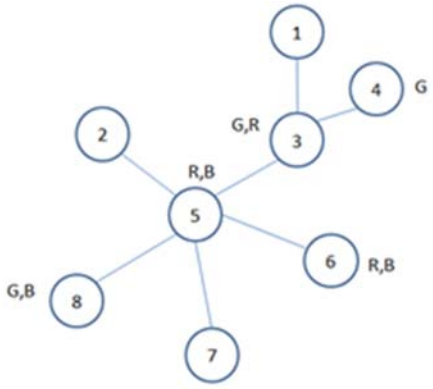
Is h_1 admissible? Is it consistent?
Is h_2 admissible? Is it consistent?

Solution

a. S-G
b. S-A-B-D-G
c. S-A-C-G
d. h_1 is neither admissible nor consistent; h_2 is admissible but is not consistent.

Problem 10

Consider the graph-coloring problem on the following tree-structured CSP.



We assume there are three available colors (R,G,B) and some nodes are constrained to use only a subset of these colors, as indicated above. Show all the steps for applying the tree-structured CSP algorithm for finding a solution to this problem.

Solution

The first step is to do a topological sort of the graph: for example, (1, 3, 4, 5, 2, 8, 7, 6). The next step is to do arc-consistency in the graph between each node and its parent, in reverse topological order. For this case, the only change the algorithm does is to the domain of node 3 (it limits it to R) and to the domain of 1 (it limits it to G, B). Therefore, the domain for each variable is 1: (G, B), 2: (R, G, B), 3:(R), 4: (G), 5: (R, B), 6: (R, B), 7: (R, G, B), 8: (G, B). The final step is to follow the topological ordering and assign any consistent value from the arc-consistent domain to each variable. The result is a consistent assignment. For example: (1: G, 3: R, 4: G, 5: B, 2: R, 8: G, 7: G, 6: R).

Problem 11

For each of the following problems, determine whether an algorithm to optimally solve the problem requires worst-case computation time that is polynomial or exponential in the parameters d and m (assuming that $P \neq NP$).

a. A map has d regions. Colors have been applied to all d regions, drawing from a set of m possible colors. Your algorithm needs to decide whether or not any two adjacent regions have the same color.
b. A map has d regions. Your algorithm needs to assign colors to all d regions, drawing colors from a set of m possible colors, in order to guarantee that no two adjacent regions have the same color.
c. Your algorithm needs to find its way out of a maze drawn on a d-by-d grid.
d. Your algorithm needs to find the shortest path in a d-by-d maze while hitting m waypoints (equivalent to dots in MP1 part 1.2).
e. Your algorithm needs to solve a planning problem in a blocks world consisting of d blocks.

Solution

a. Polynomial in d, polynomial (actually constant) in m.
b. Polynomial in m, exponential in d.
c. Polynomial in d (this is Dijkstra's).
d. Polynomial in d, exponential in m.
e. Exponential in d (or even doubly exponential in d).

Problem 12

How can randomness be incorporated into a game tree? How about partial observability (imperfect information)?

Solution

We need to add a chance node for every random event in the game, for example, a dice throw or random cards being dealt. The children of the chance node correspond to all possible outcomes, and each outcome also has a probability associated with it. The value of the chance node is computed by taking the expectation (sum of values of children nodes multiplied by their probabilities).
Partial observability gives rise to states being grouped into information sets for each player. An information set consists of all states that look the same from the viewpoint of one of the players.

Problem 13

In the lectures, we covered Nash equilibrium strategies for simultaneous move games. We can also consider minimax strategies for such games, defined in the same way as for alternating games. What would be the minimax strategies in the Prisoner's Dilemma, Stag Hunt, and Game

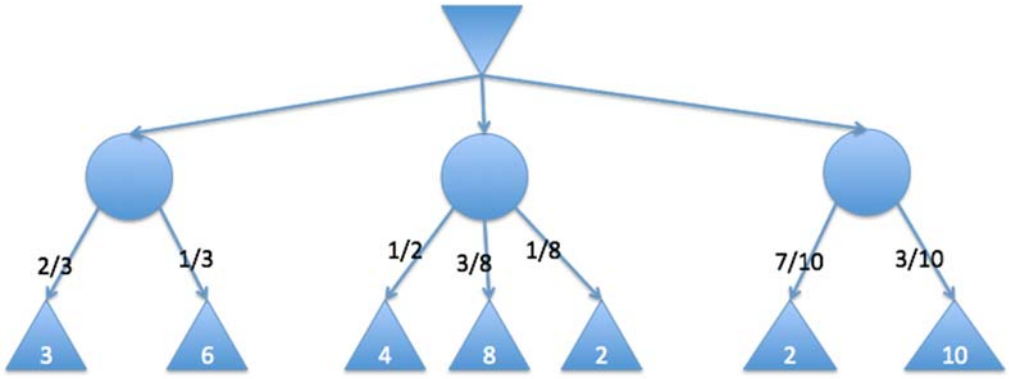
of Chicken? Do they differ from Nash equilibrium strategies? When/why would one prefer to choose a minimax strategy rather than a Nash equilibrium strategy?

Solution

A minimax strategy is a strategy that maximizes your worst-case payoff. Prisoner's Dilemma: minimax strategy is still to testify (defect). Stag Hunt: go for the hare. Game of Chicken: go chicken. You may want to use a minimax strategy if you want to avoid the need to coordinate with your opponent, or if you have reason to believe that your opponent may not be rational.

Problem 14

Consider the following expectiminimax tree:



Circle nodes are chance nodes, the top node is a min node, and the bottom nodes are max nodes.

a. For each circle, calculate the node values, as per expectiminimax definition.
b. Which action should the min player take?

Solution

a. The node values, from left to right, are as follows
   i. 2/3 * 3 + 1/3 * 6 = 4;
   ii. ½ * 4 + 3/8 * 8 + 1/8 * 2 = 5.25;
   iii. 7/10 * 2 + 3 / 10 * 10 = 4.4.
b. The leftmost action should be taken.

Problem 15

Suppose that both Alice and Bob want to go from one place to another. There are two routes R1 and R2. The utility of a route is inversely proportional to the number of cars on the road. For instance, if both Alice and Bob choose route R1, the utility of R1 for each of them is 1/2.

a. Write down the payoff matrix.
b. Is this a zero-sum game?
c. Find dominant strategies (if any).
d. Find pure strategy equilibria (if any).
e. Find the mixed strategy equilibrium.

Solution

a. The payoff matrix is as follow.

	Alice		
Bob		R1	R2
	R1	(1/2, 1/2)	(1, 1)
	R2	(1, 1)	(1/2, 1/2)

b. No.
c. There is no dominant strategy.
d. There are two pure equilibria: (R1, R2) and (R2, R1).
e. Denote the probability that Alice chooses R1 is p and Bob chooses R1 is q. Therefore, the expected payoff Alice will have via choosing R1 is
   ½ q + (1 – q);
   the expected payoff Alice will have via choosing R2 is
   q + ½ (1 – q).
   Since this mix strategy is an equilibrium, we have
   ½ q + (1 – q) = q + ½ (1 – q),
   which yields that q = 1/2 .
   Similarly, we obtain that p = ½.
   Hence, we have the mixed strategy is that both Alice and Bob choose R1 with probability ½.