

Chapter 2

Q 2.1, Q2.2, Q2.4, Q2.5 (for this question you may also need to present the diagrams for agents), Q2.6, Q 2.10, Q2.11

Ans: Check solution manual for Answers.

Chapter 3

Q 3.1 Explain why problem formulation must follow goal formulation.

Ans: In goal formulation, we decide which aspects of the world we are interested in, and which can be ignored or abstracted away. Then in problem formulation we decide how to manipulate the important aspects (and ignore the others). If we did problem formulation first we would not know what to include and what to leave out. That said, it can happen that there is a **cycle of iterations between goal formulation, problem formulation, and problem solving** until one arrives at a sufficiently useful and efficient solution.

Q 3.2 Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

- a. Formulate this problem. How large is the state space?
- b. In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now? [book 75](#)
- c. From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?
- d. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

Ans:

a. We'll define the coordinate system so that the center of the maze is at (0, 0), and the maze itself is a square from (-1, -1) to (1, 1).

Initial state: robot at coordinate (0, 0), facing North.

Goal test: either $|x| > 1$ or $|y| > 1$ where (x, y) is the current location.

Successor function: move forwards any distance d; change direction robot it facing.

Cost function: total distance moved.

The state space is infinitely large, since the robot's position is continuous.

b. The state will record the intersection the robot is currently at, along with the direction it's facing. At the end of each corridor leaving the maze we will have an exit node. We'll assume some node corresponds to the center of the maze.

Initial state: at the center of the maze facing North.

Goal test: at an exit node.

Successor function: move to the next intersection in front of us, if there is one; turn to face a new direction.

Cost function: total distance moved.

There are $4n$ states, where n is the number of intersections.

c. Initial state: at the center of the maze.

Goal test: at an exit node.

Successor function: move to next intersection to the North, South, East, or West.

Cost function: total distance moved.

We no longer need to keep track of the robot's orientation since it is irrelevant to predicting the outcome of our actions, and not part of the goal test. The motor system that executes this plan will need to keep track of the robot's current orientation, to know when to rotate the robot.

d. State abstractions:

(i) Ignoring the height of the robot off the ground, whether it is tilted off the vertical.

(ii) The robot can face in only four directions.

(iii) Other parts of the world ignored: possibility of other robots in the maze, the weather in the Caribbean.

Action abstractions:

(i) We assumed all positions we safely accessible: the robot couldn't get stuck or damaged.

(ii) The robot can move as far as it wants, without having to recharge its batteries.

(iii) Simplified movement system: moving forwards a certain distance, rather than controlled each individual motor and watching the sensors to detect collisions.

Q 3.9 The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968).

Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution.

Ans: Here is one possible representation: A state is a six-tuple of integers listing the number of missionaries, cannibals, and boats on the first side, and then the second side of the river. The goal is a state with 3 missionaries and 3 cannibals on the second side. The cost function is one per action, and the successors of a state are all the states that move 1 or 2 people and 1 boat from one side to another.

Q3.10 Define in your own words the following terms: state, state space, search tree, search node, goal, action, transition model, and branching factor.

Ans: A state is a situation that an agent can find itself in. We distinguish two types of states: world states (the actual concrete situations in the real world) and representational states (the abstract descriptions of the real world that are used by the agent in deliberating about what to do).

A state space is a graph whose nodes are the set of all states, and whose links are actions that transform one state into another.

A search tree is a tree (a graph with no undirected loops) in which the root node is the start state and the set of children for each node consists of the states reachable by taking any action.

A search node is a node in the search tree.

A goal is a state that the agent is trying to reach.

An action is something that the agent can choose to do.

A successor function described the agent's options: given a state, it returns a set of (action, state) pairs, where each state is the state reachable by taking the action.

The branching factor in a search tree is the number of actions available to the agent.

Q3.11 What's the difference between a world state, a state description, and a search node? Why is this distinction useful?

Ans: A world state is how reality is or could be. In one world state we're in Arad, in another we're in Bucharest. The world state also includes which street we're on, what's currently on the radio, and the price of tea in China. A state description is an agent's internal description of a world state. Examples are $\text{In}(\text{Arad})$ and $\text{In}(\text{Bucharest})$. These descriptions are necessarily approximate, recording only some aspect of the state.

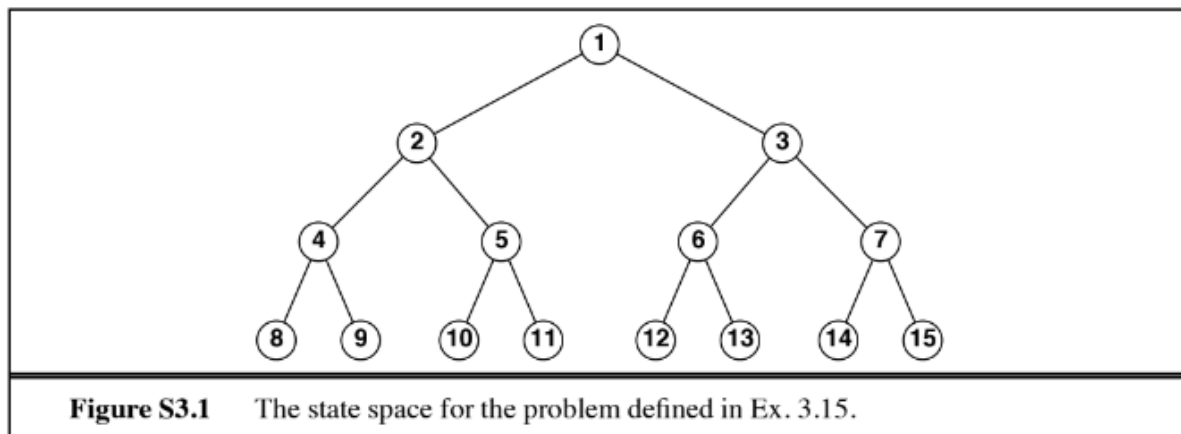
We need to distinguish between world states and state descriptions because state descriptions are lossy abstractions of the world state, because the agent could be mistaken about how the world is, because the agent might want to imagine things that aren't true but it could make true, and because the agent cares about the world not its internal representation of it. Search nodes are generated during search, representing a state the search process knows how to reach. They contain additional information aside from the state description, such as the sequence of actions used to reach this state. This distinction is

useful because we may generate different search nodes which have the same state, and because search nodes contain more information than a state representation.

Q3.15 Consider a state space where the start state is number 1 and each state k has two successors: numbers $2k$ and $2k + 1$.

- Draw the portion of the state space for states 1 to 15.
- Suppose the goal state is 11. List the order in which nodes will be visited for breadth first search, depth-limited search with limit 3, and iterative deepening search.
- How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?
- Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?
- Call the action going from k to $2k$ Left, and the action going to $2k + 1$ Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

Ans:



- See Figure S3.1.
- Breadth-first: 1 2 3 4 5 6 7 8 9 10 11
Depth-limited: 1 2 4 8 9 5 10 11
Iterative deepening: 1; 1 2 3; 1 2 4 5 3 6 7; 1 2 4 8 9 5 10 11
- Bidirectional search is very useful, because the only successor of n in the reverse direction is $\lfloor (n/2) \rfloor$. This helps focus the search. The branching factor is 2 in the forward direction; 1 in the reverse direction.

- d. Yes; start at the goal, and apply the single reverse successor action until you reach 1.
- e. The solution can be read off the binary numeral for the goal number. Write the goal number in binary. Since we can only reach positive integers, this binary expansion begins with a 1. From most- to least- significant bit, skipping the initial 1, go Left to the node $2n$ if this bit is 0 and go Right to node $2n + 1$ if it is 1. For example, suppose the goal is 11, which is 1011 in binary. The solution is therefore Left, Right, Right.

3.17 On page 90, we mentioned **iterative lengthening search**, an iterative analog of uniform cost search. The idea is to use increasing limits on path cost. If a node is generated whose path cost exceeds the current limit, it is immediately discarded. For each new iteration, the limit is set to the lowest path cost of any node discarded in the previous iteration.

- a. Show that this algorithm is optimal for general path costs.
- b. Consider a uniform tree with branching factor b , solution depth d , and unit step costs. How many iterations will iterative lengthening require?
- c. Now consider step costs drawn from the continuous range $[\epsilon, 1]$, where $0 < \epsilon < 1$. How many iterations are required in the worst case?
- d. Implement the algorithm and apply it to instances of the 8-puzzle and traveling salesperson problems. Compare the algorithm's performance to that of uniform-cost search, and comment on your results.

Ans:

- a. The algorithm expands nodes in order of increasing path cost; therefore the first goal it encounters will be the goal with the cheapest cost.
- b. It will be the same as iterative deepening, d iterations, in which $O(bd)$ nodes are generated.
- c. d/ϵ
- d. Implementation not shown.

3.18 Describe a state space in which iterative deepening search performs much worse than depth-first search (for example, $O(n^2)$ vs. $O(n)$).

Ans: Consider a domain in which every state has a single successor, and there is a single goal at depth n . Then depth-first search will find the goal in n steps, whereas iterative deepening search will take $1+2+3+\dots+n = O(n^2)$ steps.

3.21 Prove each of the following statements, or give a counterexample:

- a. Breadth-first search is a special case of uniform-cost search.
- b. Depth-first search is a special case of best-first tree search.
- c. Uniform-cost search is a special case of A* search.

Ans:

- a. When all step costs are equal, $g(n) \propto \text{depth}(n)$, so uniform-cost search reproduces breadth-first search.
- b. Breadth-first search is best-first search with $f(n) = \text{depth}(n)$; depth-first search is best-first search with $f(n) = -\text{depth}(n)$; uniform-cost search is best-first search with $f(n) = g(n)$.
- c. Uniform-cost search is A* search with $h(n)=0$.

3.27 n vehicles occupy squares $(1, 1)$ through $(n, 1)$ (i.e., the bottom row) of an $n \times n$ grid. The vehicles must be moved to the top row but in reverse order; so the vehicle i that starts in $(i, 1)$ must end up in $(n - i + 1, n)$. On each time step, every one of the n vehicles can move one square up, down, left, or right, or stay put; but if a vehicle stays put, one other adjacent vehicle (but not more than one) can hop over it. Two vehicles cannot occupy the same square.

- a. Calculate the size of the state space as a function of n .
- b. Calculate the branching factor as a function of n .
- c. Suppose that vehicle i is at (x_i, y_i) ; write a nontrivial admissible heuristic h_i for the number of moves it will require to get to its goal location $(n - i + 1, n)$, assuming no other vehicles are on the grid.
- d. Which of the following heuristics are admissible for the problem of moving all n vehicles to their destinations? Explain.
 - (i) $\sum_{i=1}^n h_i$.
 - (ii) $\max\{h_1, \dots, h_n\}$.
 - (iii) $\min\{h_1, \dots, h_n\}$.

Ans:

- a. n^{2n} . There are n vehicles in n^2 locations, so roughly (ignoring the one-per-square constraint) $(n^2)^n = n^{2n}$ states.
- b. 5^n .
- c. Manhattan distance, i.e., $|(n - i + 1) - x_i| + |n - y_i|$. This is exact for a lone vehicle.
- d. Only (iii) $\min\{h_1, \dots, h_n\}$. The explanation is nontrivial as it requires two observations. First, let the *work* W in a given solution be the total *distance* moved by all vehicles over their joint trajectories; that is, for each vehicle, add the lengths of all the steps taken. We have $W \geq \sum_i h_i \geq n \cdot \min\{h_1, \dots, h_n\}$. Second, the total work we can get done per step is $\leq n$. (Note that for every car that jumps 2, another car has to stay put (move 0), so the total work per step is bounded by n .) Hence, completing all the work requires at least $n \cdot \min\{h_1, \dots, h_n\} / n = \min\{h_1, \dots, h_n\}$ steps.

3.28 Invent a heuristic function for the 8-puzzle that sometimes overestimates, and show how it can lead to a suboptimal solution on a particular problem. (You can use a computer to help if you want.) Prove that if h never overestimates by more than c , A^* using h returns a solution whose cost exceeds that of the optimal solution by no more than c .

Ans:

3.28 The heuristic $h = h_1 + h_2$ (adding misplaced tiles and Manhattan distance) sometimes overestimates. Now, suppose $h(n) \leq h^*(n) + c$ (as given) and let G_2 be a goal that is suboptimal by more than c , i.e., $g(G_2) > C^* + c$. Now consider any node n on a path to an optimal goal. We have

$$\begin{aligned}
 f(n) &= g(n) + h(n) \\
 &\leq g(n) + h^*(n) + c \\
 &\leq C^* + c \\
 &\leq g(G_2)
 \end{aligned}$$

so G_2 will never be expanded before an optimal goal is expanded.

3.29 Prove that if a heuristic is consistent, it must be admissible. Construct an admissible heuristic that is not consistent.

Ans:

3.29 A heuristic is consistent iff, for every node n and every successor n' of n generated by any action a ,

$$h(n) \leq c(n, a, n') + h(n')$$

One simple proof is by induction on the number k of nodes on the shortest path to any goal from n . For $k = 1$, let n' be the goal node; then $h(n) \leq c(n, a, n')$. For the inductive

case, assume n' is on the shortest path k steps from the goal and that $h(n')$ is admissible by hypothesis; then

$$h(n) \leq c(n, a, n') + h(n') \leq c(n, a, n') + h^*(n') = h^*(n)$$

so $h(n)$ at $k + 1$ steps from the goal is also admissible.