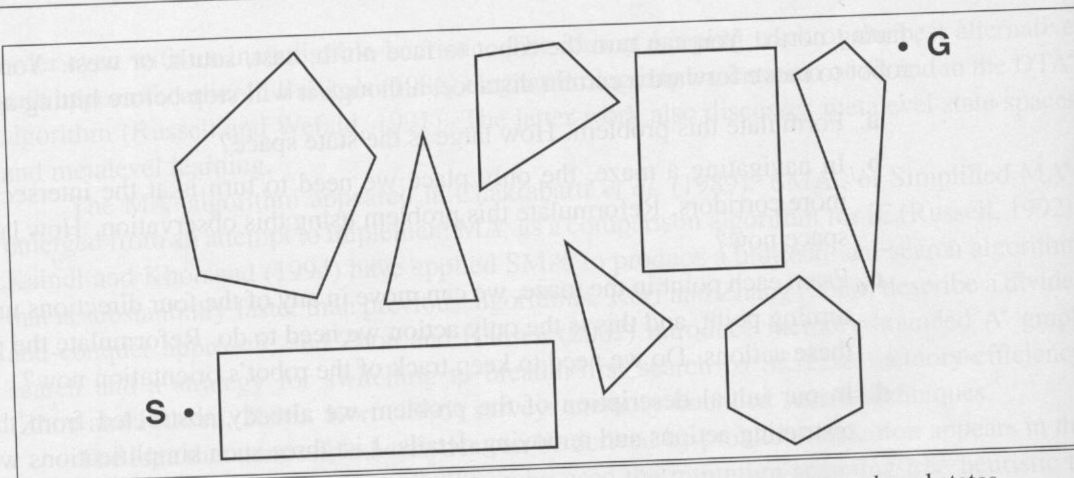


**Figure 3.31** A scene with polygonal obstacles. $S$ and $G$ are the start and goal states.

**c.** You have a program that outputs the message "illegal input record" when fed a certain file of input records. You know that processing of each record is independent of the other records. You want to discover what record is illegal.

**d.** You have three jugs, measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure out exactly one gallon.

**3.7** Consider the problem of finding the shortest path between two points on a plane that has convex polygonal obstacles as shown in Figure 3.31. This is an idealization of the problem that a robot has to solve to navigate in a crowded environment.

**a.** Suppose the state space consists of all positions $(x, y)$ in the plane. How many states are there? How many paths are there to the goal?

**b.** Explain briefly why the shortest path from one polygon vertex to any other in the scene must consist of straight-line segments joining some of the vertices of the polygons. Define a good state space now. How large is this state space?

**c.** Define the necessary functions to implement the search problem, including an ACTIONS function that takes a vertex as input and returns a set of vectors, each of which maps the current vertex to one of the vertices that can be reached in a straight line. (Do not forget the neighbors on the same polygon.) Use the straight-line distance for the heuristic function.

**d.** Apply one or more of the algorithms in this chapter to solve a range of problems in the domain, and comment on their performance.

**3.8** On page 68, we said that we would not consider problems with negative path costs. In this exercise, we explore this decision in more depth.

**a.** Suppose that actions can have arbitrarily large negative costs; explain why this possibility would force any optimal algorithm to explore the entire state space.

**b.** Does it help if we insist that step costs must be greater than or equal to some negative constant $c$? Consider both trees and graphs.

**c.** Suppose that a set of actions forms a loop in the state space such that executing the set in some order results in no net change to the state. If all of these actions have negative cost, what does this imply about the optimal behavior for an agent in such an environment?

**d.** One can easily imagine actions with high negative cost, even in domains such as route finding. For example, some stretches of road might have such beautiful scenery as to far outweigh the normal costs in terms of time and fuel. Explain, in precise terms, within the context of state-space search, why humans do not drive around scenic loops indefinitely, and explain how to define the state space and actions for route finding so that artificial agents can also avoid looping.

**e.** Can you think of a real domain in which step costs are such as to cause looping?

**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).

**a.** Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.

**b.** Implement and solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?

**c.** Why do you think people have a hard time solving this puzzle, given that the state space is so simple?

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

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

**3.12** An action such as *Go(Sibiu)* really consists of a long sequence of finer-grained actions: turn on the car, release the brake, accelerate forward, etc. Having composite actions of this kind reduces the number of steps in a solution sequence, thereby reducing the search time. Suppose we take this to the logical extreme, by making super-composite actions out of every possible sequence of *Go* actions. Then every problem instance is solved by a single super-composite action, such as *Go(Sibiu)Go(Rimnicu Vilcea)Go(Pitesti)Go(Bucharest)*. Explain how search would work in this formulation. Is this a practical approach for speeding up problem solving?

**3.13** Prove that GRAPH-SEARCH satisfies the graph separation property illustrated in Figure 3.9. (*Hint*: Begin by showing that the property holds at the start, then show that if it holds before an iteration of the algorithm, it holds afterwards.) Describe a search algorithm that violates the property.

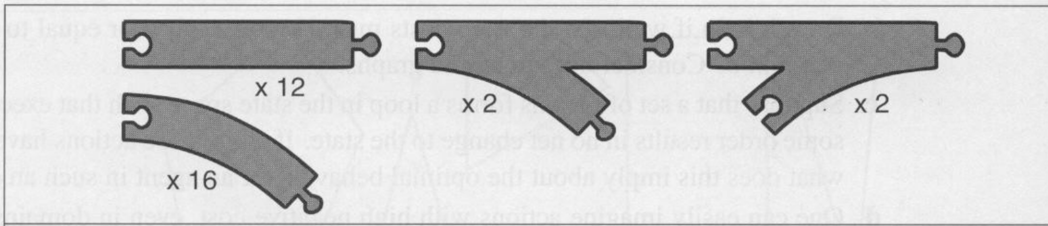


**Figure 3.32** The track pieces in a wooden railway set; each is labeled with the number of copies in the set. Note that curved pieces and "fork" pieces ("switches" or "points") can be flipped over so they can curve in either direction. Each curve subtends 45 degrees.

**3.14** Which of the following are true and which are false? Explain your answers.

**a.** Depth-first search always expands at least as many nodes as A* search with an admissible heuristic.

**b.** $h(n) = 0$ is an admissible heuristic for the 8-puzzle.

**c.** A* is of no use in robotics because percepts, states, and actions are continuous.

**d.** Breadth-first search is complete even if zero step costs are allowed.

**e.** Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.

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

**a.** Draw the portion of the state space for states 1 to 15.

**b.** 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.

**c.** How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?

**d.** 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?

**e.** 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?

**3.16** A basic wooden railway set contains the pieces shown in Figure 3.32. The task is to connect these pieces into a railway that has no overlapping tracks and no loose ends where a train could run off onto the floor.

**a.** Suppose that the pieces fit together *exactly* with no slack. Give a precise formulation of the task as a search problem.

**b.** Identify a suitable uninformed search algorithm for this task and explain your choice.

**c.** Explain why removing any one of the "fork" pieces makes the problem unsolvable.