

1.

- a. **Initial state:** No regions colored.

Goal test: All regions colored, and no two adjacent regions have the same color.

Successor function: Assign a color to a region.

Cost function: Number of assignments

- b. **Initial state:** jugs have values [0, 0, 0].

Successor function: given values $[x, y, z]$, generate $[12, y, z]$, $[x, 8, z]$, $[x, y, 3]$ (by filling); $[0, y, z]$, $[x, 0, z]$, $[x, y, 0]$ (by emptying); or for any two jugs with current values x and y , pour y into x ; this changes the jug with x to the minimum of $x + y$ and the capacity of the jug, and decrements the jug with y by the amount gained by the first jug.

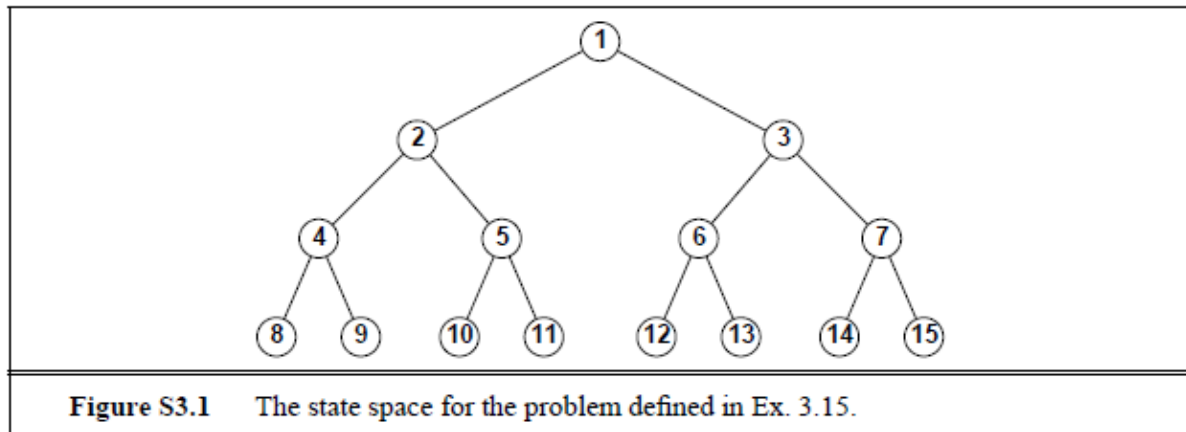
Cost function: Number of actions

2.

- a. If we consider all (x, y) points, then there are an infinite number of states, and of paths.

- b. (For this problem, we consider the start and goal points to be vertices.) The shortest distance between two points is a straight line, and if it is not possible to travel in a straight line because some obstacle is in the way, then the next shortest distance is a sequence of line segments, end-to-end, that deviate from the straight line by as little as possible. So the first segment of this sequence must go from the start point to a tangent point on an obstacle – any path that gave the obstacle a wider girth would be longer. Because the obstacles are polygonal, the tangent points must be at vertices of the obstacles, and hence the entire path must go from vertex to vertex. So now the state space is the set of vertices, of which there are 35

3.



- a. See Figure S3.1.
- b. 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
- c. 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.
- 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.