**Homework 4**

**2. Chapter 3, Exercise 8:**

1. **Initial State:** plain map

**Actions:** Color a region with one of the four colors

**Transition Model:** The map is updated with the colored region

**Goal Test:** No two adjacent regions have the same color, and all regions are colored

**Path Cost:** The number of the times of coloring the regions

1. **Initial State:** A room with a 3-foot-tall monkey, some bananas suspended from the 8-foot ceiling, and two stackable, movable, climbable 3-foot-high crates

**Actions:** Move crate, stack crate, climb crate, grab bananas, and jump down from the crate

**Transition Model:** The state of the room, including the location of the monkey and crate positions, is updated based on actions.

**Goal Test:** The monkey gets bananas

**Path Cost:** The number of times of climbing, moving, stacking the crates, and jumping down

1. **Initial State:** Three jugs measuring 12 gallons, 8 gallons, and 3 gallons, and an empty water faucet

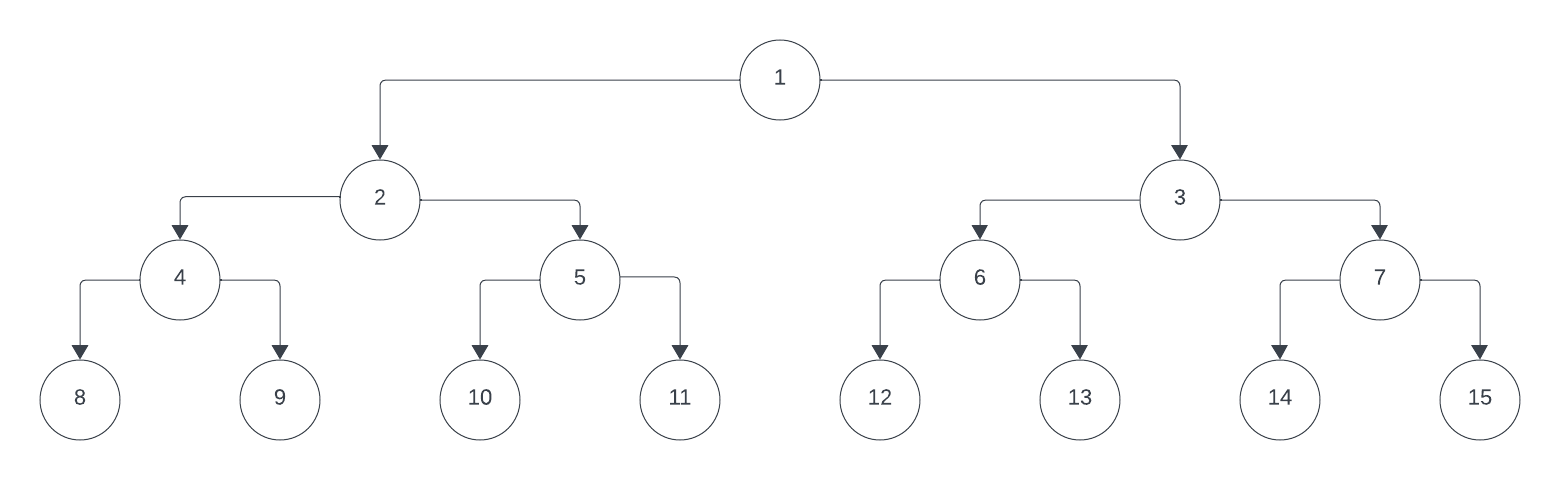
**Actions:** Fill a jug, empty a jug, and pour from one jug to another or onto the ground

**Transition Model:** Update the amount of water in of each jug

**Goal State:** One of the jugs contains 1 gallon of water

**Path Cost:** The number of times of filling, emptying, and pouring a jug

**3. Chapter 3, Exercise 18:**



1. breadth-first search: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

depth-limited search with limit 3: 1, 2, 4, 8, 9, 5, 10, 11

iterative deepening search: (1), (1, 2, 3), (1, 2, 4, 5, 3, 6, 7), (1, 2, 4, 8, 9, 5, 10, 11)

1. The bidirectional search is efficient in this problem because there is only one successor function (n/2) in the reverse direction. So, the branching factor is 1 in the reverse direction and 2 in the forward direction.
2. Yes. The path of the goal can be identified reversely by starting from the goal state and using the reverse successor function (n/2) until reaching the initial state.
3. By analyzing the binary representation of the goal state, we start from the second leftmost bit. If it is 0, go left (n/2). Otherwise, go right (2n+1). For example, the binary representation of the goal state 11 is 1011. So, the sequence of actions is left, right, and right.

**4. Evaluation Function:**

1. It emulates the best-first search algorithm. By setting h(n) = -g(n), we choose the nodes with the lowest path cost g(n). This is equivalent to the best-first search because it expands the node that is the closest to the goal based on the path length.
2. It emulates the uniform cost search. By making the heuristic value equal to the current path cost, we choose the nodes based on their true path cost. This is the same as the uniform cost search, where nodes with the lowest true path cost are explored first, aiming to find the lowest-cost path to the goal.

**5. Chapter 3, Exercise 25:**

1. Breadth-first search is a special case of uniform-cost search when all step costs between states are equal. In BFS, all edges have the same cost, so UCS is essentially BFS with edge costs as the priority. So, when all step costs are equal, BFS is a special case of UCS.
2. Depth-first search is a special case of best-first tree search with f(n) = - depth(n). So, it expands the deepest nodes.
3. Uniform-cost search is a special case of A\* search when h(n) is equal to zero for all nodes. In UCS, only the actual path cost g(n) is considered but A\* search generalizes UCS by incorporating h(n). When h(n) is zero for all nodes, A\* behaves the same as UCS, making UCS a special case of A\* search with a heuristic that always evaluates to zero.

**6. Chapter 3, Exercise 27:**

Lugoj = 0 + 244 = 244

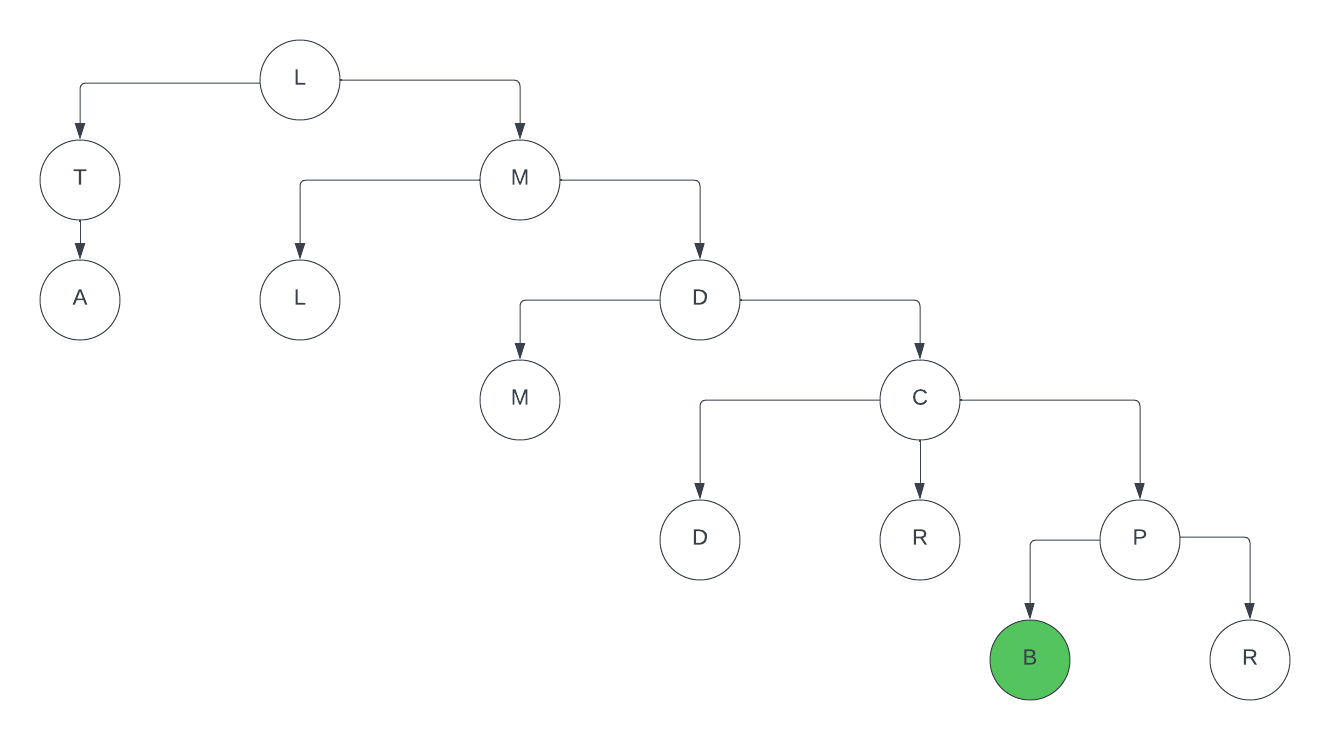
Timisoara = 111 + 329 = 440 , Mehadia = 70 + 241 = 311

Lugoj = (70 + 70) + 244 = 384 , Dobreta = (70 + 75) + 242 = 387

Mehadia = (311 + 75) + 241 = 627 , Craiova = (145 + 120) + 160 = 425

Rimnicu Vilcea = (265 + 146) + 193 = 604 , Pitesti = (265 + 138) + 100 = 503 , Dobreta = (265 + 120) + 242 = 627 , Arad (After Lugoj) = (111 + 118) + 329 = 558

**Bucharest = (403 + 101) + 0 = 504** , Rimnicu Vilcea = (403 + 97) + 193 = 693



**7. Chapter 3, Exercise 38:**

1. The straight-line distance between two cities represents the shortest possible path between them. On the other hand, the MST heuristic connects all cities in a tree, which may involve longer distances than the straight-line distances. So, MST has higher values than (dominates) the straight-line distance.