Sheet 3 Solution

3.1 Explain why problem formulation must follow goal formulation.

Solution:

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.

3.6 Give a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.

a. Using only four colors, you have to color a planar map in such a way that no two adjacent regions have the same color.

Solution:

- 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. A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two stackable, movable, climbable 3-foot-high crates.

Solution:

- Initial state: As described in the text.
- Goal test: Monkey has bananas.
- Successor function: Hop on crate; Hop off crate; Push crate from one spot to another; Walk from one spot to another; grab bananas (if standing on crate).
- Cost function: Number of actions.

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.

   Solution:

   - Initial state: considering all input records.
   - Goal test: considering a single record, and it gives “illegal input” message.
   - Successor function: run again on the first half of the records; run again on the second half of the records.
   - Cost function: Number of runs. Note: This is a contingency problem; you need to see whether a run gives an error message or not to decide what to do next.

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.

   Solution:

   - 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.

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

- 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** [**transition model**] 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.

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

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 in (Arad) and in(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 description is lossy abstractions of the world state, because the agent could be mistaken about [illegible] 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.