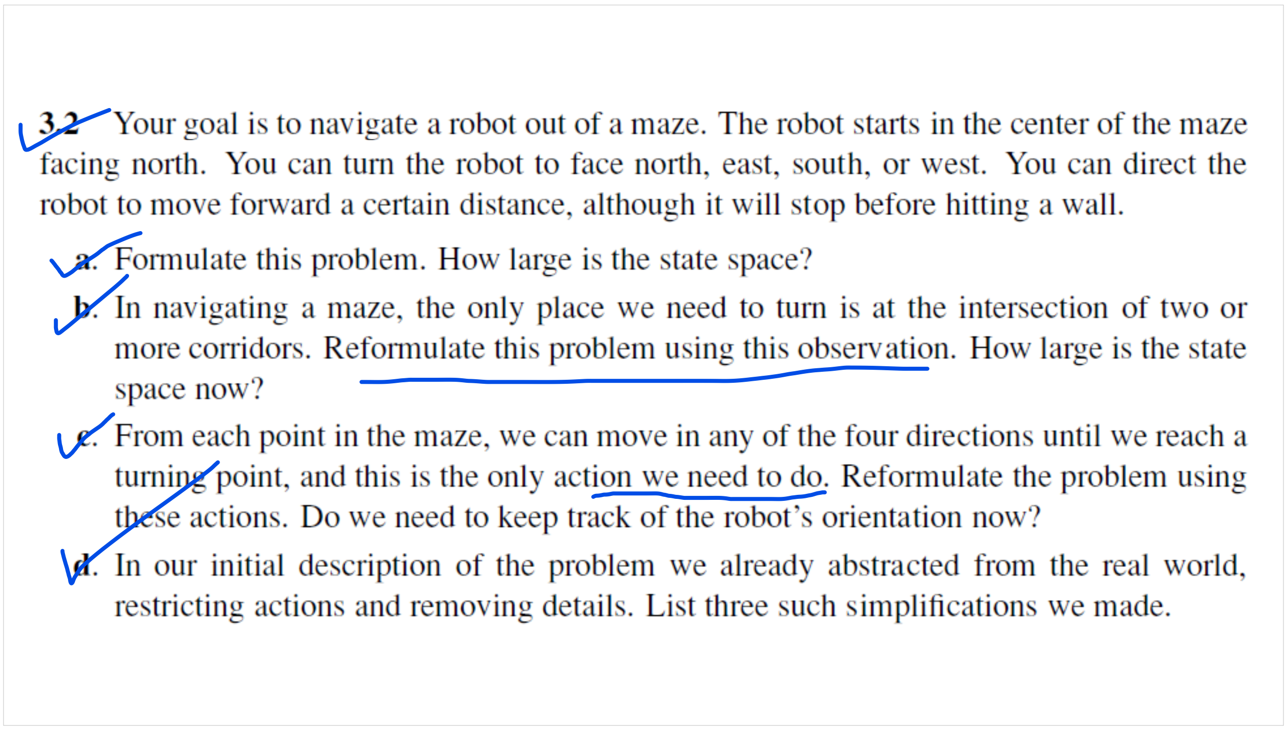
Uninformed Search

solve it again alone baa.

Chapter 3



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?

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.

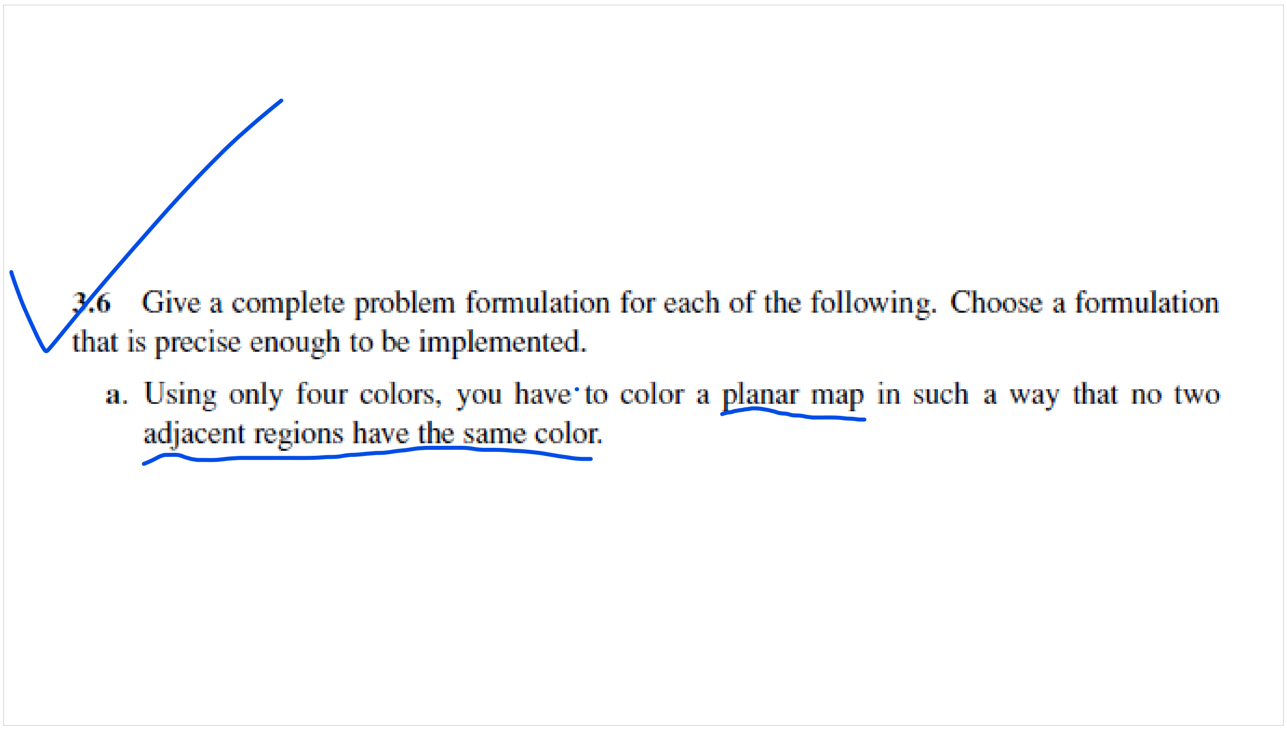
A.
Initial State: (x_o, y_o, r_o) which represents the initial position and rotation of the robot
Goal Test: (x,y) not inside the maze
Successor Function: Given (x,y,r), the successors are (x,y,0), (x,y,90), (x,y,180), (x,y,270), (x+d*cos(r), y+d*sin(r), r) where d is the certain distance
Cost: 1 per action

To compute the state space size, we divide the space into square of size dxd (where d is the distance traveled in each step). The state space size = number of empty square * 4 possible rotation.

B.
Successor Function: Given (x,y,r), the successor is (x+d*cos(r), y+d*sin(r), r) where d is the certain distance, and if (x,y) is an intersection, add (x,y,0), (x,y,90), (x,y,180), (x,y,270) to the successors
The state space size = (number of empty square - number of intersections) * 2 possible rotations + number of intersections * 4 possible rotations.

C.
Successor Function: Given (x,y,r), the successors are (x,y,0), (x,y,90), (x,y,180), (x,y,270), (x+L*cos(r), y+L*sin(r), r) where L is the distance to the upcoming intersection in direction r
The state space size = number of intersections * 4 possible rotations.

D.
1- The robot will accurately face the requested direction (no rotation error).
2- The robot will accurately move d distance in every action.
3- We discretized the action space.



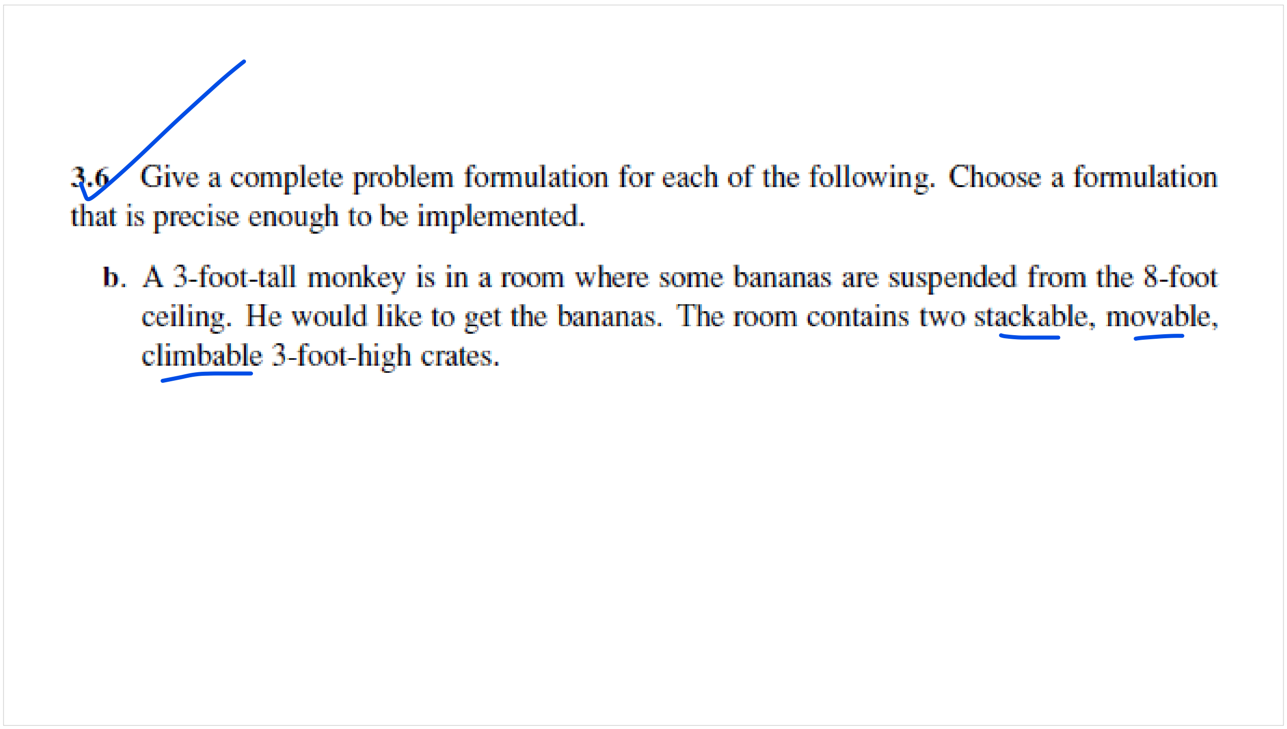
3.6 Give a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.

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.

Initial state: State = [None, None, ..] which is a list containing a color for each node in the map
Goal test: State[n] != None for all node n in the map.
Actions: Color(n, c) where n is an uncolored node and c is a color not used by any node adjacent to n.
Transition Model:
Given the state “State”, and Action “Color(n, c)”, the successor “Successor” is defined:

```
Successor[i]=c           if i = n
Successor[i]=State[i]    if i != n
```

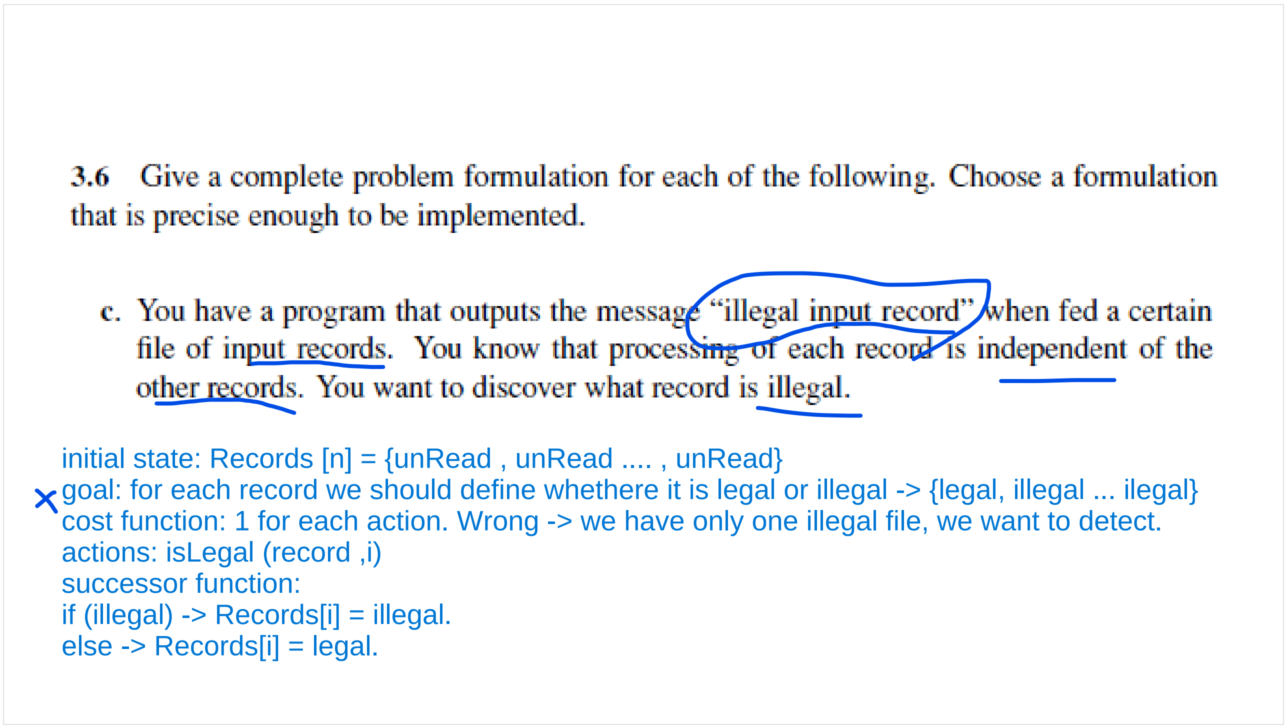
Cost function: 1 for each action. If we want to find the color assignment with the least number of used color, we could make the cost for using a new color higher



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

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.

The main idea of the solution:
Initial state: as described in the question.
Goal test: you have banana.
Actions: open any box you have the key for, get the contents of any open box.
Cost function: number of actions.
*** We will solve a similar problem when we get to Classical Planning



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

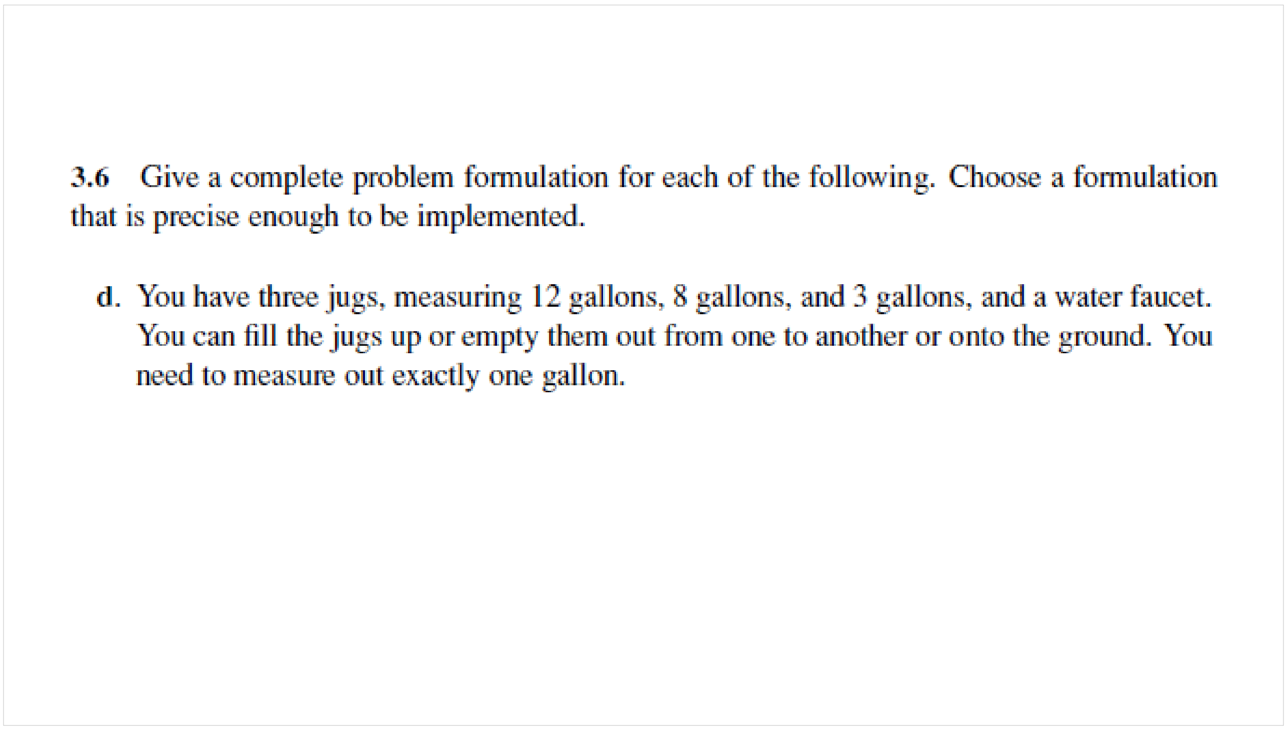
c. You have a program that outputs the message "illegal input record" when fed a certain file of input records. You know that processing of each record is independent of the other records. You want to discover what record is illegal.

initial state: Records [n] = {unRead , unRead .... , unRead}
× goal: for each record we should define whethere it is legal or illegal -> {legal, illegal ... ilegal}
cost function: 1 for each action. Wrong -> we have only one illegal file, we want to detect.
actions: isLegal (record ,i)
successor function:
if (illegal) -> Records[i] = illegal.
else -> Records[i] = legal.

Initial state: State = All records are in the suspected list
Goal: Number of records in the suspected list = 1
Action: Send subset S of suspected records to the program
Transition Model:

```
        Given the output of the program to the subset S we send
                If result is "illegal input record":
                        Successor = S being the new suspected list
                Else:
                        Successor = all the suspected records in "State" except the
                ones in S
```

Cost: 1 per action



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

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.

Initial state: Content = [0, 0, 0] (all jugs are empty)
Goal: There exists j where Content[j] = 1
Actions:

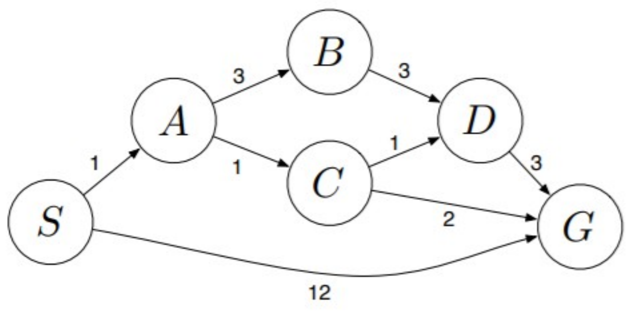
- Fill jug j from faucet where content[j] < capacity[j]
- Empty jug j into jug k where j != k, content[j] >0 and content[k] < capacity[k]
- Empty jug j onto the ground where content[j] > 0

Successor:

```
    If action = Empty jug j onto the ground:
        newcontent[j] = 0 // jug was emptied
    If action = Empty jug j into jug k
        amount = min( content[j], capacity[k] - content[k] )
        newcontent[j] = content[j] - amount
        newcontent[k] = content[k] + amount
    If action = Fill jug j from faucet:
        newcontent[j] = capacity[j]
```

Cost: 1 for each action (or 1 + amount of water drawn from faucet to discourage water wasting. We add 1 since we cannot have a cost of 0).

Apply BFS, DFS, and UCS to the following problem:



BFS:
Path: S->G
Expansion: S,A,G

DFS:
Path: S->A->B->D->G
Expansion: S, A, B, D, G

UCS:
Path: S->A->C->G
Expansion: S, A, C, D, B, G