Q1.

2.4 For each of the following activities, give a PEAS description of the task environment and characterize it in terms of the properties listed in Section 2.3.2.

- Playing soccer.
- Exploring the subsurface oceans of Titan.
- Shopping for used AI books on the Internet.
- Playing a tennis match.

1. Partially observe, stochastic, sequential, dynamic, continuous, multi-agent
2. Partially observe, stochastic, sequential, dynamic, continuous, single-agent
3. Partially observe, deterministic, sequential static, discrete, single-agent
   This can be single-agent
4. fully observe, stochastic, episodic, dynamic continuous, multi-agent

Q2.

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.

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.

(a). Initial state: No colored region
Goal state: No two adjacent regions are the same color and all regions colored
Successor function: color a region
Cost function: Number of actions.

(b). Initial state: A 3-foot-tall monkey in a room with bananas suspended from the 8-foot ceiling
Goal state: Monkey gets bananas
Successor function: Put on crates, Put off crates, Put

crates from one spot to another, monkey grab from one spot to another, get bananas.

Cost function: Number of actions

Q3.

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.

(a). state: a three-tuple of integers. like

(1, 2, 1)

the first is the number of missionaries in first side, the second is the number of cannibals in first side, the third is whether the boat is in left side (1 or 0)

Goal: 3 missionaries and 3 cannibals on the second side

Cost function: 1 per action

Successors of a state: move 1 or 2 people and 1 boat from one side to another.

Q4.

1. The *n-queens* problem is to place *n queens* on an n-by-n chessboard, such that no queen attacks another, as per chess rules. Pose the problem as a search problem.

The start state is no queens on an n-by-n chess board, and the goal state is n queens on the chessboard with no queen attacks another. The tasks is to make a sequence of placements, such that the agent ends up being

a goal state

Q5.

3. In the *rabbit leap problem,* three east-bound rabbits stand in a line blocked by three west-bound rabbits. They are crossing a stream with stones placed in the east west direction in a line. There is one empty stone between them.

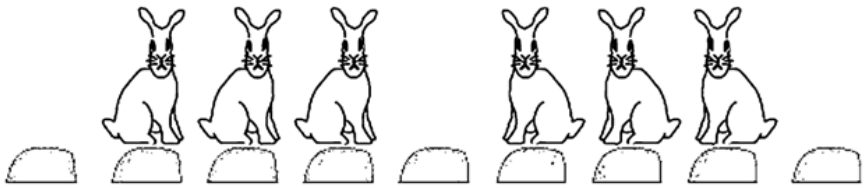
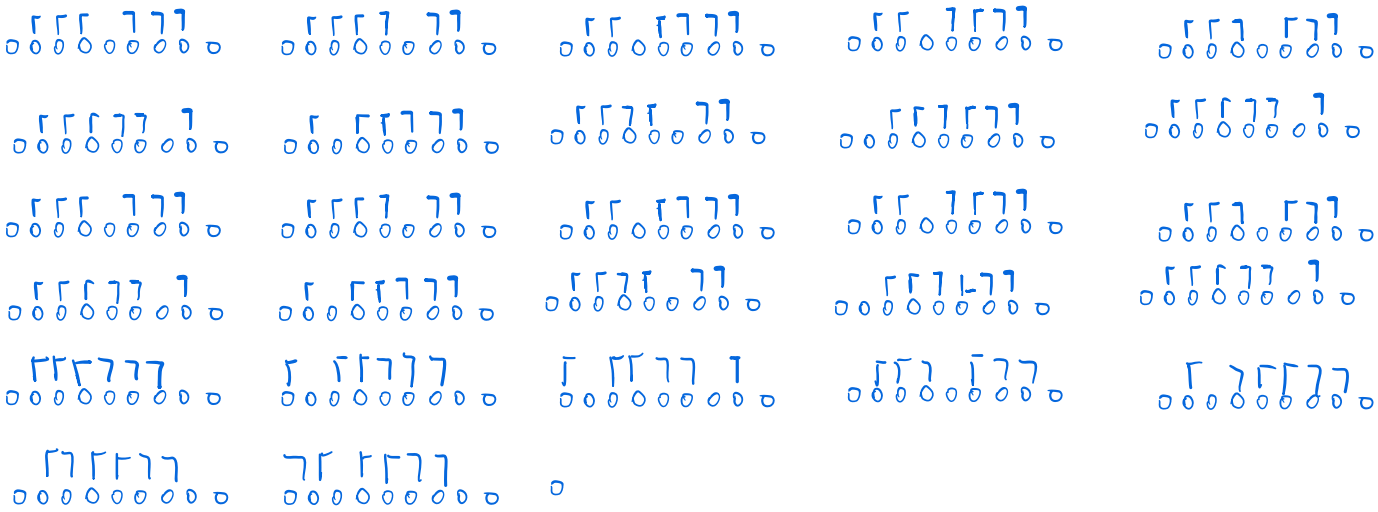


FIGURE 2.32 Rabbits waiting to cross. Each rabbit can jump over one, but not more than that. How can they avoid getting into a deadlock?

The rabbits can only move forward one step or two steps. They can jump over one rabbit if the need arises, but not more than that. Are they smart enough to cross each other without having to step into the water? Draw the state space for solving the problem, and find the solution path in the state space graph.



Q6.

10. Given the *moveGen* function in the table below, and the corresponding state space graph, the task is to find a path from the start node *S* to the goal node *J*.

```
   moveGen
S → (D C B A)
A → (S B J E)
B → (S F A)
C → (S D H G)
D → (S I C)
E → (A J K)
F → (B K J)
G → (C L)
H → (C I M L)
I → (D H)
J → (A F E)
K → (E F)
L → (G H M)
M → (L H)
```

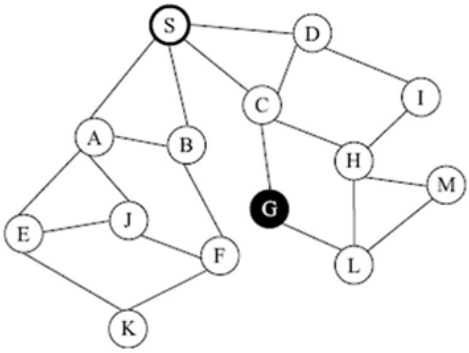


FIGURE 2.34 A small search problem. The task is to find a path from the start node *S* to the goal node *J*.

Show the *OPEN* and *CLOSED* lists for the *DFS* and the *BFS* algorithms. What is the path found by each of them?

DFS :

OPEN:
(S, Nil)
(A,S)(B,S)(C,S)(D,S)
(I,D)(A,S)(B,S)(C,S)
(H,I)(A,S)(B,S)(C,S)
(M,H)(L,H)(A,S)(B,S)(C,S)
(G,C)(A,S)(B,S)
(F,B) (A,S)
(K,F)(J,F)(A,S)

CLOSED:
()
(S,Nil)
(D,S)(S,Nil)
(I,D)(D,S)(S,Nil)
(H,I)(I,D)(D,S)(S,Nil)
(M,H)(L,H)(C,S)(H,I)(I,D)(D,S)(S,Nil)
(G,C)(B,S)(M,H)(L,H)(C,S)(H,I)(I,D)(D,S)(S,Nil)
(F,B)(G,C)(B,S)(M,H)(L,H)(C,S)(H,I)(I,D)(D,S)(S,Nil)

The back points are
J->F,F->B,B->S

BFS :

OPEN:
(S,Nil)
(A,S)(B,S)(C,S)(D,S)
(E,A)(J,A)(B,S)(C,S)(D,S)

CLOSED:
()
(S,Nil)
(A,S)(S,Nil)

The back points are
J->A,A->S

Q7.

7. **Sudoku**: see www.websudoku.com. Consider using Search to solve Sudoku puzzles: You are given a partially filled grid to start, already know there is an answer. *(credits CMU)* [25 Points]

a. Define a state representation for Sudoku answer search. A state is a partially filled, valid grid in which no rows, column, or 3x3 square contains duplicated digits.

b. Write a pseudocode for the successor function (generate a list of successor states from the current state) given your representation. You can assume you have functions like "duplicated" that returns true if there are two identical non-empty elements (or non-zero digits) in a collection.

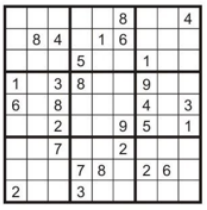


Figure 1: A sample Sudoku puzzle with 28 digits filled initially.

c. Write a pseudocode for goal function that returns true if a state is a goal. You can assume the state is reached by the successor function above so it is a valid state.

d. If the puzzle begins with 28 digits filled, what is *L*, the length of the shortest path to the goal using your representation?

(a). Start state: a partially filled grid
Goal state: All grid filled and no rows, column or 3×3 square contains duplicated digits
Successor function: fill a number
Cost function: Number of actions.

(b) fillNum(State): Take a state as input and returns a set of states that are reachable in one step from the input state.
goalTest(state): Returns true if the input state is the

goal state and false otherwise.

duplicated (state): returns true if there are two identical non-empty elements (or non-zero digits) in a collection.

```
Generate and Test ( )
  while goal Test (state) is false:
      for states in till Num (state ):
          if duplicated (states) is false:
              state += states
  return state.
```

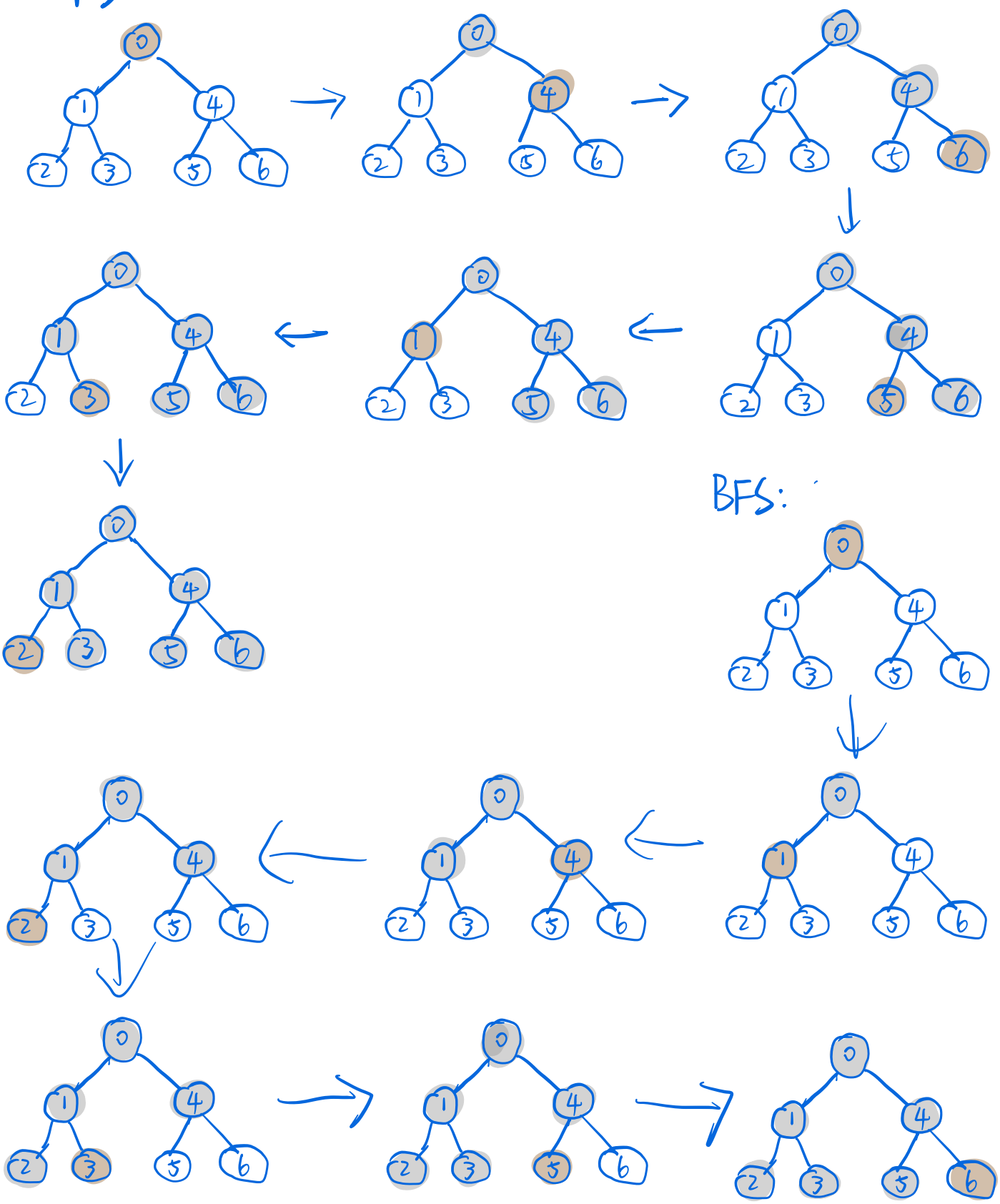
(c) duplicated (state): returns true if there are two identical non-empty elements (or non-zero digits) in a collection.

```
goal Test (state)
 for columns in state:
        if duplicated (state):
            return False
 for rows in state:
        if duplicated (state):
            return False
 for 3x3 squares in state:
        if duplicated (state):
            return False

 return True.
```

(d) 53 (every number is filled without duplicated at the first time)

Q8. DFS:



BFS:

CODE :

```
graph = {
    '0': ['1', '4'],
    '1': ['2', '3'],
    '2': ['1'],
    '3': ['1'],
    '4': ['5', '6'],
    '5': ['4'],
    '6': ['4'],
}

def BFS(graph, s):
    queue = []
    queue.append(s)
    seen = set()
    seen.add(s)
    while len(queue) > 0:
        vertex = queue.pop(0)
        nodes = graph[vertex]
        for w in nodes:
            if w not in seen:
                queue.append(w)
                seen.add(w)
        print(vertex)

def DFS(graph, s):
    stack = []
    stack.append(s)
    seen = set()
    seen.add(s)
    while len(stack) > 0:
        vertex = stack.pop()
        nodes = graph[vertex]
        for w in nodes:
            if w not in seen:
                stack.append(w)
                seen.add(w)
        print(vertex)
print('BFS')
print(BFS(graph, '0'))
print('DFS')
print(DFS(graph, '0'))
```

Answer:

BFS	DFS
0	0
1	4
4	6
2	5
3	1
5	3
6	2
None	None