AI Tutorial Problems

-Dr. Radhika V. Kulkarni

Associate Professor, Dept. of Computer Engineering,

Vishwakarma Institute of Technology, Pune.

Sources:

1. Stuart Russell & Peter Norvig, "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.
2. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill
3. Deepak Khemani, “A First Course in Artificial Intelligence”, McGraw Hill
4. Saroj Kaushik, “Artificial Intelligence”, Cengage Publication.

DISCLAIMER

This presentation is created as a reference material for the students of TY-CS, VIT (AY 2023-24 Sem-1).

It is restricted only for the internal use and any circulation is strictly prohibited.

Water Jug Problem

Sources:

1. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill.

Water Jug Problem (1)

- Problem Description: We are given two jugs, a 4-gallon one and 3-gallon one. Neither has any measuring marked on it. There is a pump, which can be used to fill the jugs with water. How can we get exactly 2 gallons of water into 4-gallon jug?

- Solution:
- The state space for this problem can be described as the set of ordered pairs of integers (X, Y) such that X = 0, 1, 2, 3 or 4 and Y = 0, 1, 2 or 3; X is the number of gallons of water in the 4-gallon jug and Y the quantity of water in the 3-gallon jug.
- The start state is (0, 0) and the goal state is (2, n) for any value of n, as the problem does not specify how many gallons need to be filled in the 3-gallon jug (0, 1, 2, 3). So, the problem has one initial state and many goal states. Some problems may have many initial states and one or many goal states.
- As provided in the problem statement, at any given state we can do either of the following operations:
  1. Fill a jug
  2. Empty a jug
  3. Transfer water from one jug to another until either of them gets completely filled or empty.

Water Jug Problem (2)

- The operators to be used to solve the problem can be described as shown in Fig. 2.3:

1	(x, y) if $x < 4$	→	$(4, y)$	Fill the 4-gallon jug
2	(x, y) if $y < 3$	→	$(x, 3)$	Fill the 3-gallon jug
3	(x, y) if $x > 0$	→	$(x - d, y)$	Pour some water out of the 4-gallon jug
4	(x, y) if $y > 0$	→	$(x, y - d)$	Pour some water out of the 3-gallon jug
5	$(x, y\}$ if $x > 0$	→	$(0, y)$	Empty the 4-gallon jug on the ground
6	(x, y) if $y > 0$	→	$(x, 0)$	Empty the 3-gallon jug on the ground
7	(x, y) if $x + y \geq 4$ and $y > 0$	→	$(4, y - (4 - x))$	Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full
8	(x, y) if $x + y \geq 3$ and $x > 0$	→	$(x - (3 - y), 3)$	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full
9	(x, y) if $x + y \leq 4$ and $y > 0$	→	$(x + y, 0)$	Pour all the water from the 3-gallon jug into the 4-gallon jug
10	(x, y) if $x + y \leq 3$ and $x > 0$	→	$(0, x + y)$	Pour all the water from the 4-gallon jug into the 3-gallon jug
11	$(0, 2)$	→	$(2, 0)$	Pour the 2 gallons from the 3-gallon jug into the 4-gallon jug
12	$(2, y)$	→	$(0, y)$	Empty the 2 gallons in the 4-gallon jug on the ground

Fig. 2.3 *Production Rules for the Water Jug Problem*

Water Jug Problem (3)

- To solve the water jug problem, all we need, in addition to given problem description, is a control structure which loops through a simple cycle in which some rule whose left side matches the current state is chosen, the appropriate change to the state is made as described in the corresponding right side and the resulting state is checked to see if it corresponds to a goal state.
- The loop continues as long as it does not lead to the goal. The speed with which the problem is solved depends upon the mechanism, control structure, which is used to select the next operation.
- There are several sequences of operators which will solve the problem, two such sequences are shown here:
- **Solution1:**

Number of Steps	*Rules applied*	*4-g jug*	*3-g jug*	
1	**Initial State**	0	0	
2	R2 {Fill 3-g jug}	0	3	
3	R7 {Pour all water from 3 to 4-g jug }	3	0	
4	R2 {Fill 3-g jug}	3	3	
5	R5 {Pour from 3 to 4-g jug until it is full}	4	2	
6	R3 {Empty 4-gallon jug}	0	2	
7	R7 {Pour all water from 3 to 4-g jug}	2	0	**Goal State**

Water Jug Problem (4)

- **Solution2**:

Number of steps	*Rules applied*	*4-g jug*	*3-g jug*	
1	**Initial State**	0	0	
2	R1 {Fill 4-gallon jug}	4	0	
3	R6 {Pour from 4 to 3-g jug until it is full }	1	3	
4	R4 {Empty 3-gallon jug}	1	0	
5	R8 {Pour all water from 4 to 3-gallon jug}	0	1	
6	R1 {Fill 4-gallon jug}	4	1	
7	R6 {Pour from 4 to 3-g jug until it is full}	2	3	
8	R4 {Empty 3-gallon jug}	2	0	**Goal State**

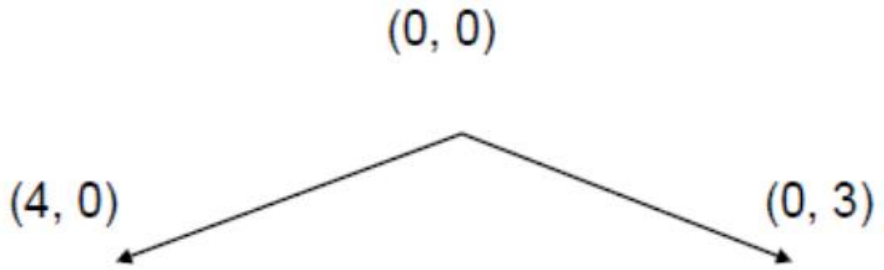
- There may be more than one solutions.

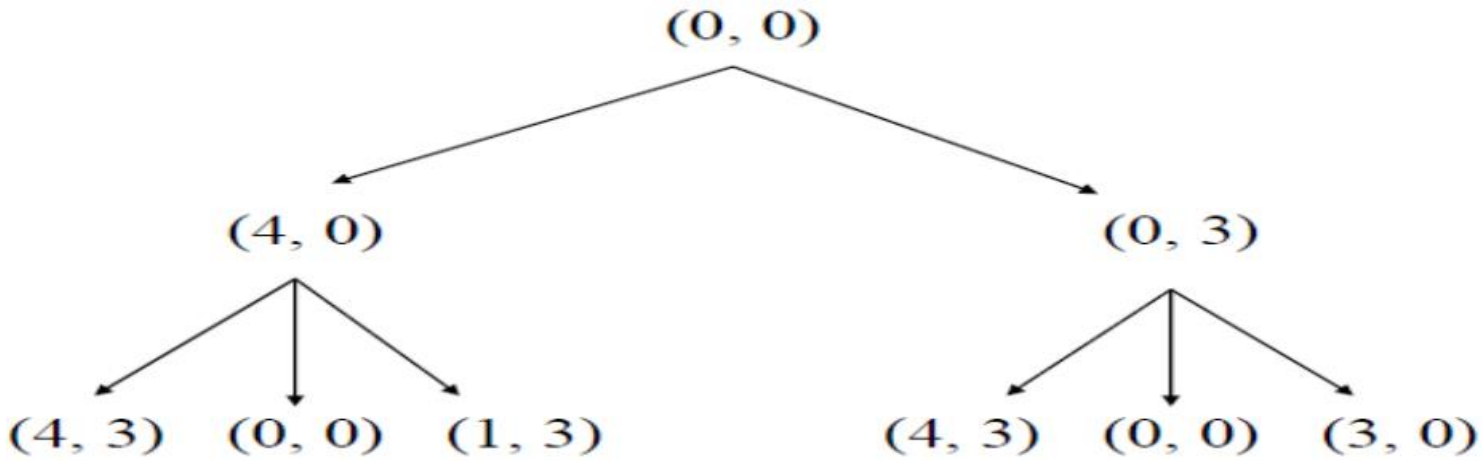
X	Y	Rule applied (Control strategy)
0	0	
4	0	I-
1	3	8
1	0	6
0	1	10
4	1	1
2	3	8

Water Jug Problem (5)

- BFS Algorithm for water jug problem:

- Construct a tree with the initial state of the problem as its root.
- Generate all the offspring of the root by applying each of the applicable rules to the initial state.
- For each leaf node, generate all its successors by applying all the rules that are appropriate.
- Repeat this process till we find a solution, if it exists.

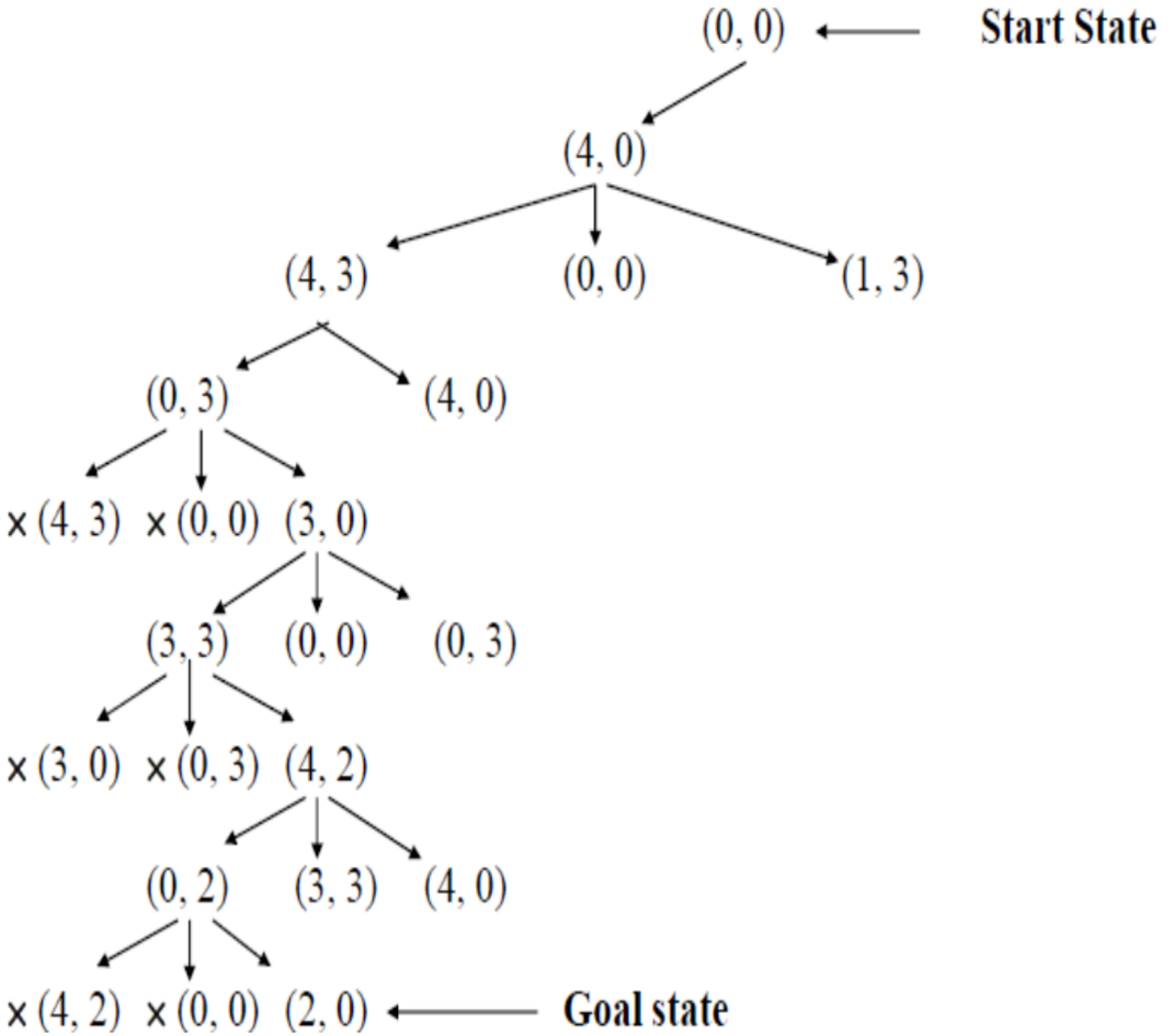




Water Jug Problem (6)

- DFS Algorithm for water jug problem:

- Here we pursue a single branch of the tree until it yields a solution or some pre-specified depth has reached.
- If solution is not found then
  - go back to immediate previous node and
  - explore other branches in DF fashion.
- Let us see the tree formation for water jug problem using DFS



Missionaries and Cannibals Problem

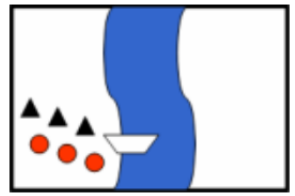
Source:

1. Stuart Russell & Peter Norvig, "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.
2. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill.

Missionaries and Cannibals Problem (1)

- Problem Description: Three missionaries and three cannibals want to cross a river. There is a boat on their side of the river that can be used by either one or two persons. How should they use this boat to cross the river in such a way that cannibals never outnumber missionaries on either side of the river? If the cannibals ever outnumber the missionaries (on either bank) then the missionaries will be eaten. How can they all cross over without anyone being eaten?

- Solution:
- The initial state is shown to the right here, where black triangles represent missionaries and red circles represent cannibals.



- Goal: Move all the missionaries and cannibals across the river.
- Constraint: Missionaries can never be outnumbered by cannibals on either side of river, or else the missionaries are killed.
- State: configuration of missionaries and cannibals and boat on each side of river.
- Operators/ Actions: Move boat containing some set of occupants across the river (in either direction) to the other side.

Missionaries and Cannibals Problem (2)

- We should make a graph search which traverse the graph from initial state and find out the final state in fewest moves. There are many AI searches that search the graphs like Breadth first search, Depth first search, or iterative deepening search.

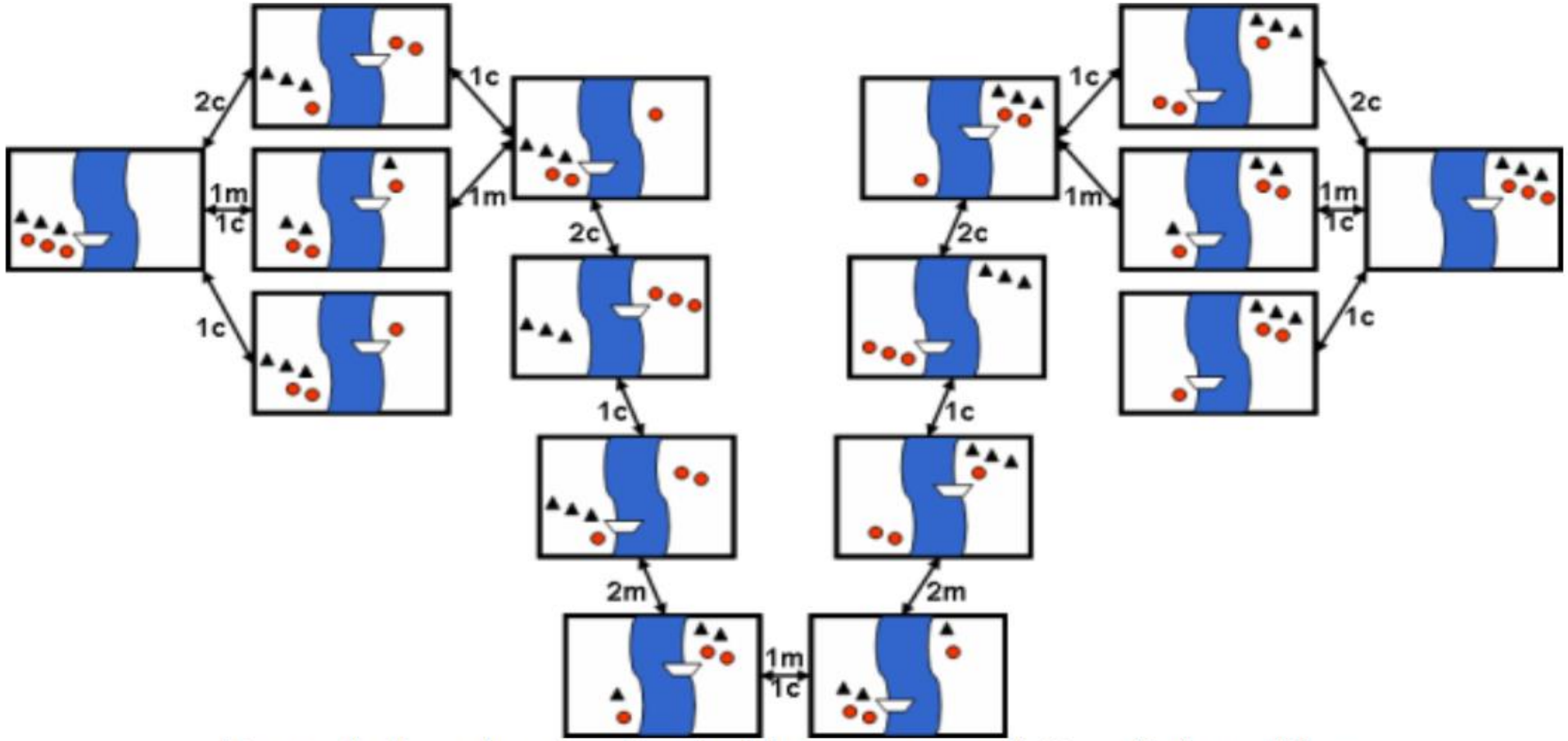


Figure 1: Search-space for the Missionaries and Cannibals problem

Missionaries and Cannibals Problem (3)

- Problem space for this problem can be described as the set of ordered pairs of left and right bank of the river as (L, R) where each bank is represented as a list State(no_of_missionaries, no_of_cannibals, presence_of_the_boat) i.e.[nM, mC, B]
  - n is the number of missionaries M, m is the number of cannibals C, and B represents boat.

- Start state: (L, R) = ([3M, 3C, 1B], [0M, 0C, 0B]) or (L, R) = ([3,3,1], [0,0,0])
  - 1B means that boat is present and 0B means it is not there on the bank of river.

- Goal state: (L, R) = ([0M, 0C, 0B], [3M, 3C, 1B]) or (L, R) = ([0,0,0], [3,3,1])

- Any state: ($[n_1M, m_1C, 1B], [n_2 M, m_2 C, 0B]$) ,

  with constraints/conditions as $n_1 (\neq 0) \geq m_1$; $n_2 (\neq 0) \geq m_2$; $n_1 + n_2 = 3$, $m_1 + m_2 = 3$.

- Note: By no means, this representation is unique. In fact, one may have number of different representations for the same problem.

Missionaries and Cannibals Problem (4)

- Production rules for Missionaries and Cannibals problem:

RN	Left side of rule	→	Right side of rule
	Rules for boat going from left bank to right bank of the river		
L1	([n_1M, m_1C, 1B], [n_2M, m_2C, 0B])	→	([(n_1-2)M, m_1C, 0B], [(n_2+2)M, m_2C, 1B])
L2	([n_1M, m_1C, 1B], [n_2M, m_2C, 0B])	→	([(n_1-1)M,(m_1-1)C,0B],[(n_2+1)M,(m_2+1)C, 1B])
L3	([n_1M, m_1C, 1B], [n_2M, m_2C, 0B])	→	([n_1M, (m_1-2)C, 0B], [n_2M, (m_2+2)C, 1B])
L4	([n_1M, m_1C, 1B], [n_2M, m_2C, 0B])	→	([(n_1-1)M, m_1C,0B],[(n_2+1)M, m_2C, 1B])
L5	([n_1M, m_1C, 1B], [n_2M, m_2C, 0B])	→	([n_1M, (m_1-1)C, 0B], [n_2M, (m_2+1)C, 1B])
	Rules for boat coming from right bank to left bank of the river		
R1	([n_1M, m_1C, 0B], [n_2M, m_2C, 1B])	→	([(n_1+2)M, m_1C, 1B], [(n_2-2)M, m_2C, 0B])
R2	([n_1M, m_1C, 0B], [n_2M, m_2C, 1B])	→	([(n_1+1)M,(m_1+1)C,1B],[(n_2-1)M,(m_2-1)C, 0B])
R3	([n_1M, m_1C, 0B], [n_2M, m_2C, 1B])	→	([n_1M, (m_1+2)C, 1B], [n_2M, (m_2-2)C, 0B])
R4	([n_1M, m_1C, 0B], [n_2M, m_2C, 1B])	→	([(n_1+1)M, m_1C,1B],[(n_2-1)M, m_2C, 0B])
R5	([n_1M, m_1C, 0B], [n_2M, m_2C, 1B])	→	([n_1M, (m_1+1)C, 1B], [n_2M, (m_2-1)C, 0B])

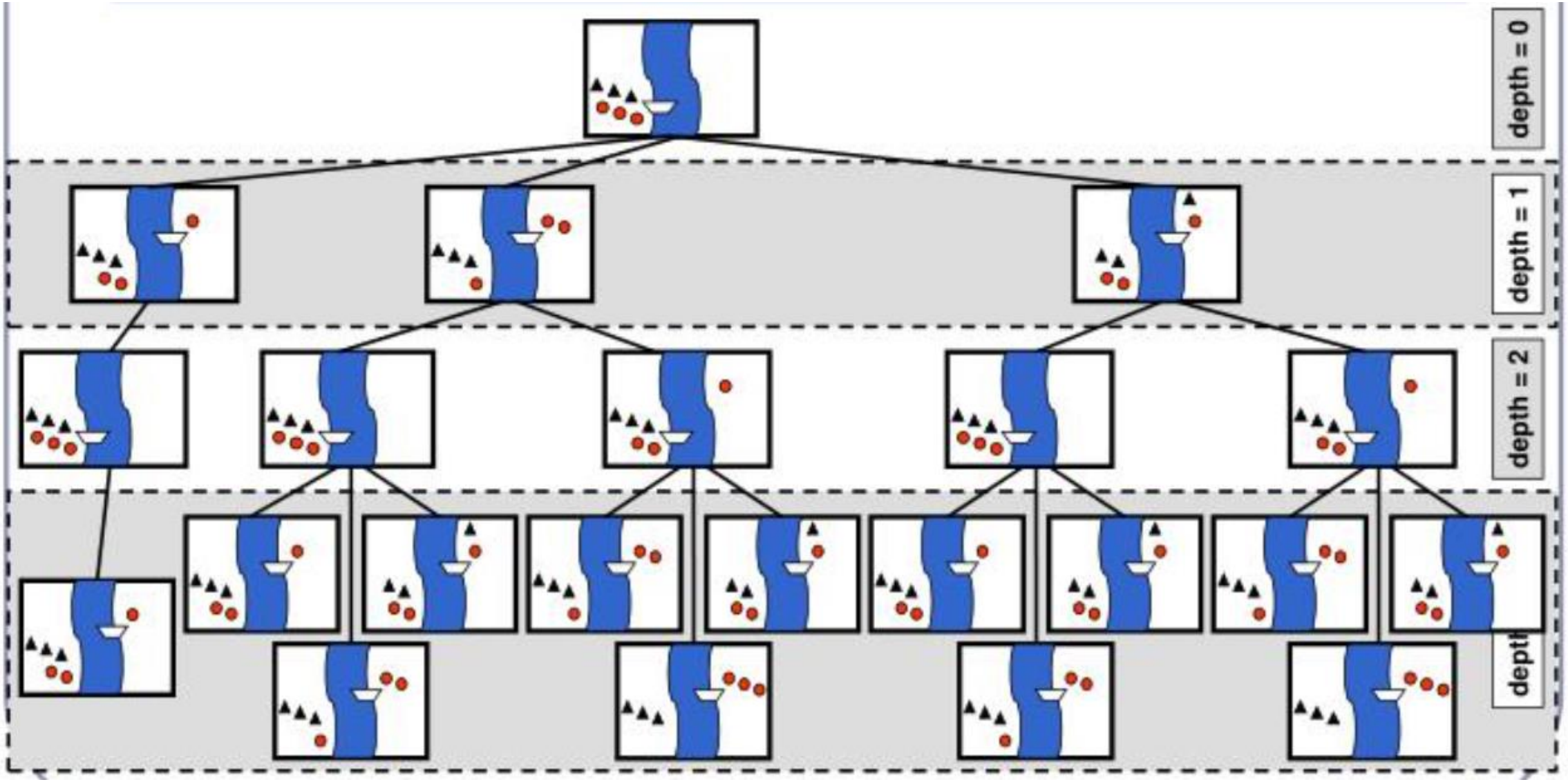
Missionaries and Cannibals Problem (5)

One of the possible paths

Start →	**([3M, 3C, 1B], [0M, 0C, 0B])**	
L2:	([2M, 2C, 0B], [1M, 1C, 1B])	**1M,1C →**
R4:	([3M, 2C, 1B], [0M, 1C, 0B])	**1M ←**
L3:	([3M, 0C, 0B], [0M, 3C, 1B])	**2C →**
R4:	([3M, 1C, 1B], [0M, 2C, 0B])	**1C ←**
L1:	([1M, 1C, 0B], [2M, 2C, 1B])	**2M →**
R2:	([2M, 2C, 1B], [1M, 1C, 0B])	**1M,1C ←**
L1:	([0M, 2C, 0B], [3M, 1C, 1B])	**2M →**
R5:	([0M, 3C, 1B], [3M, 0C, 0B])	**1C ←**
L3:	([0M, 1C, 0B], [3M, 2C, 1B])	**2C →**
R5:	([0M, 2C, 1B], [3M, 1C, 0B])	**1C ←**
L3:	**([0M, 0C, 0B], [3M, 3C, 1B])**	**2C → Goal state**

Missionaries and Cannibals Problem (5)

- BFS algorithm for Missionaries and Cannibals Problem:



8-Puzzle Problem

Source:

1. Stuart Russell & Peter Norvig, "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.
2. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill.

8-Puzzle Problem (1)

- Problem Description: The 8-puzzle is a 3 × 3 array containing eight square pieces, numbered 1 through 8, and one empty space. A piece/tile can be moved horizontally or vertically into the empty space, in effect exchanging the positions of the piece and the empty space.
- There are four possible moves, UP (move the blank space up), DOWN, LEFT and RIGHT. The aim of the game is to make a sequence of moves that will convert the board from the start state into the goal state. This example can be solved by the operator sequence UP, RIGHT, UP, LEFT, DOWN.

Initial State

2	8	3
1	6	4
7		5

Goal State

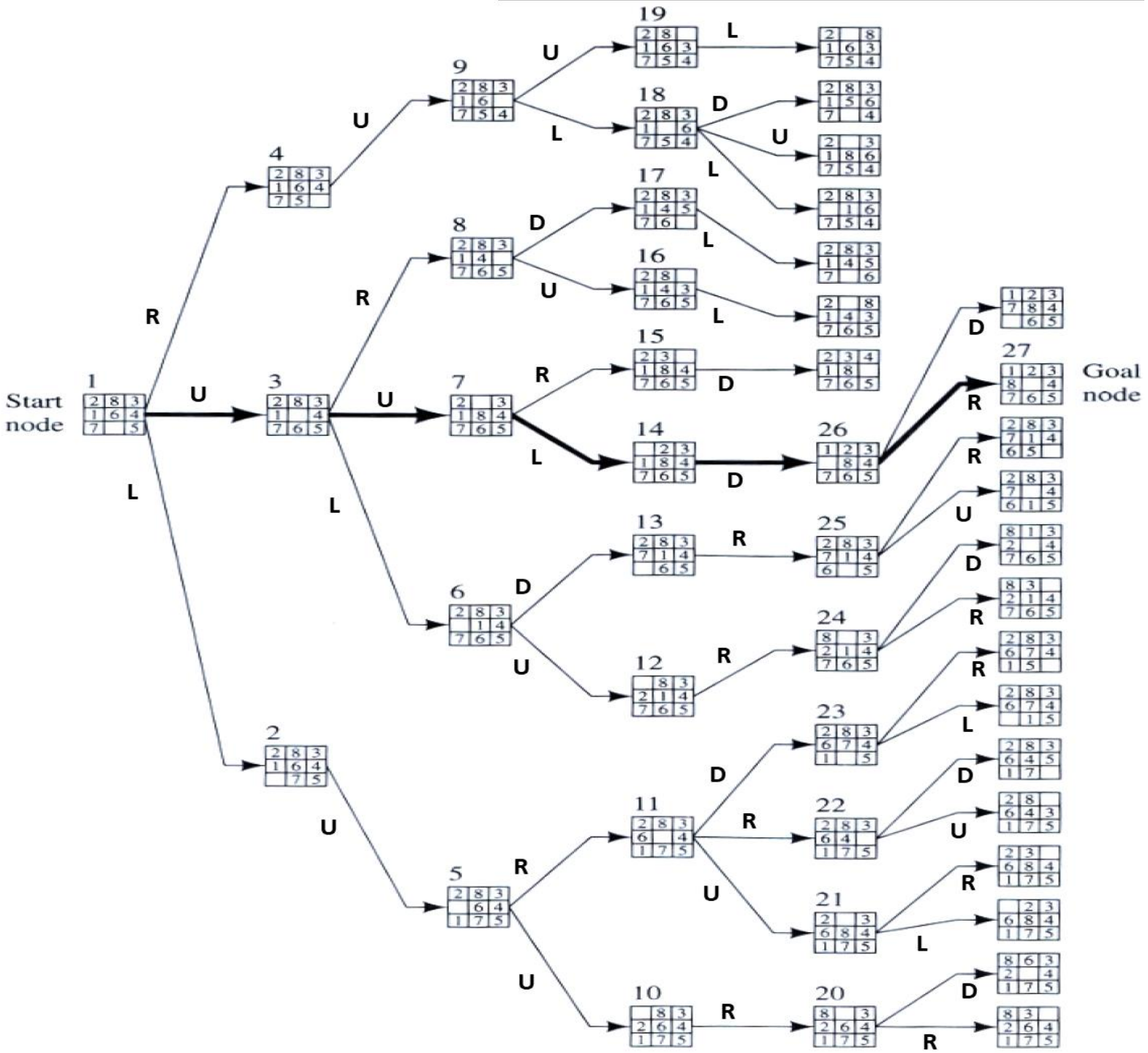
1	2	3
8		4
7	6	5

- Solution: Without heuristic it will explore each and every state using uninformed search techniques. However, applying heuristics it can explore lesser number of states and get optimal solution.
- Different heuristics:
  1. h= No. of misplaced tiles by comparing current state and goal state; explore the node with the least heuristic value.
  2. h= Sum of Euclidian distances of the tiles from their goal positions.(Euclidian distance= $\sqrt{(x_1-x_2)^2+ (y_1-y_2)^2}$).
  3. h= Sum of Manhattan distances of the tiles from their goal positions. (Manhatten distance=$|x_1-x_2|+| y_1-y_2|$)
  4. h= Number of tiles out of row + Number of tiles out of column.

8-Puzzle Problem (2)

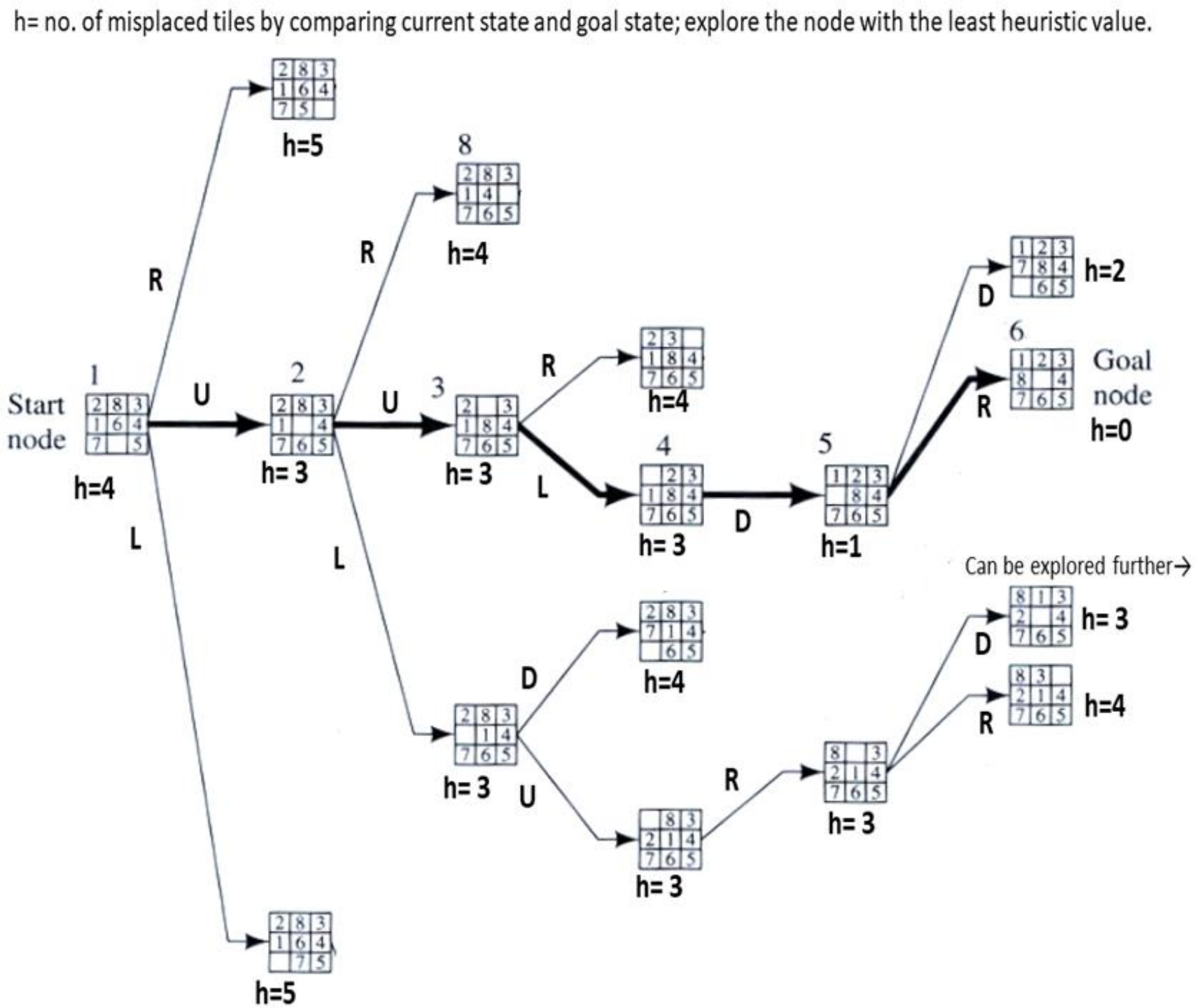
- **BFS for 8-Puzzle Problem:**

(without heuristics)



8-Puzzle Problem (3)

- **Heuristics for 8-Puzzle Problem:**
- h= no. of misplaced tiles by comparing current state and goal state;
- Explore the node with the least heuristic value.



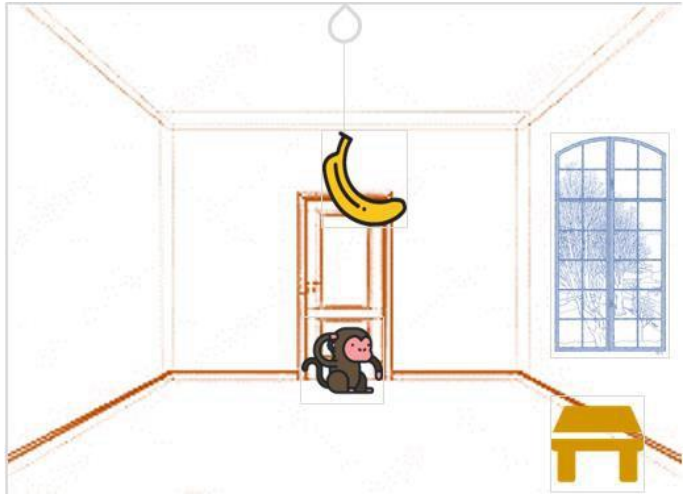
Monkey Banana Problem

Source:

1. Stuart Russell & Peter Norvig, "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.
2. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill.
3. http://people.uncw.edu/narayans/courses/csc434/monkey.pdf

Monkey Banana Problem (1)

- Problem Description: A hungry monkey is in a room, and he is near the door. The monkey is on the floor. Bananas have been hung from the center of the ceiling of the room. There is a block (or chair) present in the room near the window. The monkey wants the banana but cannot reach it.
- The monkey and box have height Low, but if the monkey climbs onto the box he will have height High, the same as the bananas.
- The actions available to the monkey include Go from one place to another, Push an object from one place to another, ClimbUp onto or ClimbDown from an object, and Grasp or Ungrasp an object.
- The result of a Grasp is that the monkey holds the object if the monkey and object are in the same place at the same height.



Monkey Banana Problem (2)

- If the monkey is clever enough, he can come to the block, drag the block to the center, climb on it, and get the banana. Below are few observations in this case −
  - Monkey can reach the block, if both of them are at the same level. From the above image, we can see that both the monkey and the block are on the floor.
  - If the block position is not at the center, then monkey can drag it to the center.
  - If monkey and the block both are on the floor, and block is at the center, then the monkey can climb up on the block. So, the vertical position of the monkey will be changed.
  - When the monkey is on the block, and block is at the center, then the monkey can get the bananas.

For more details refer to: http://people.uncw.edu/narayans/courses/csc434/monkey.pdf

Thank you!