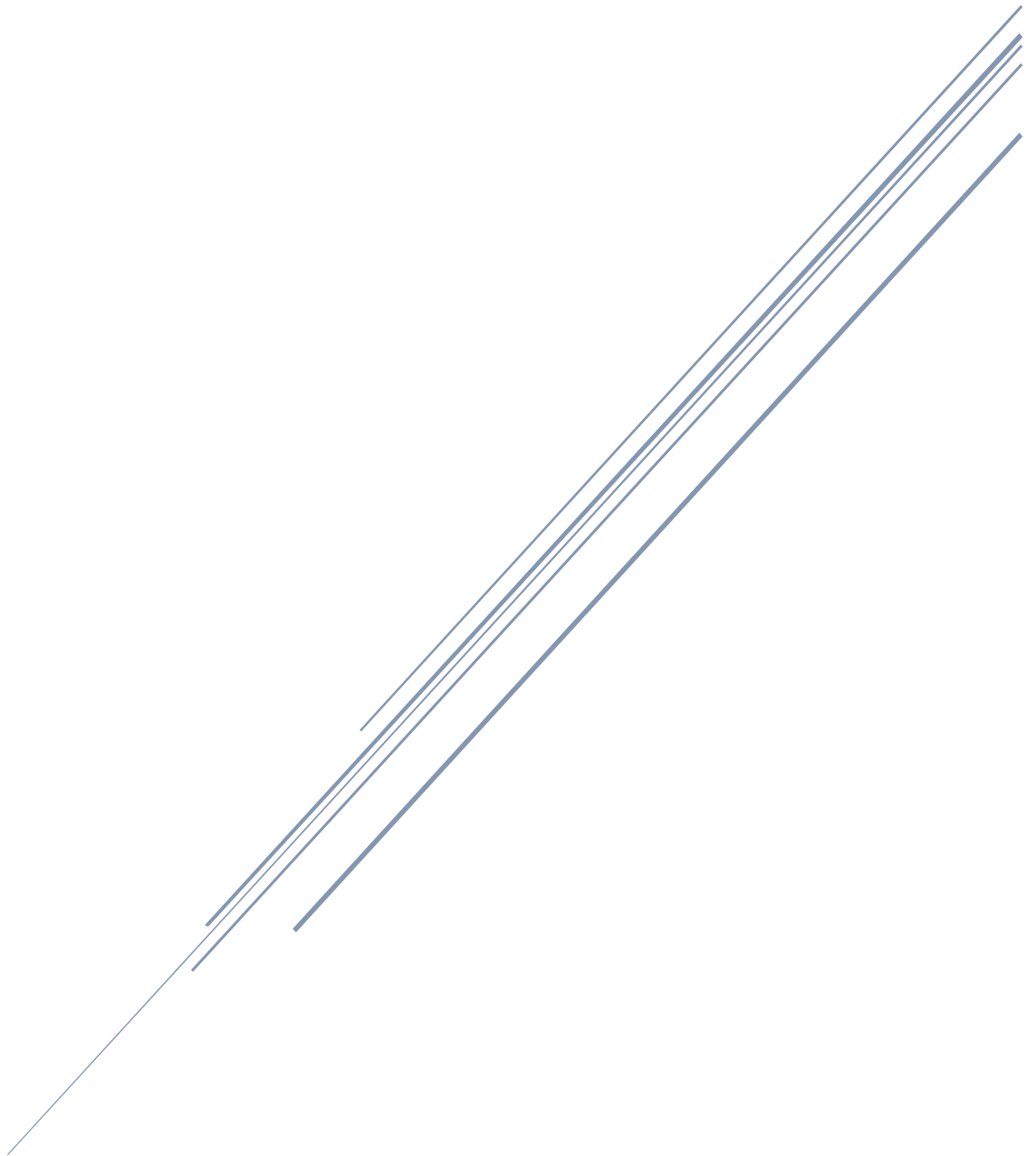


# CHAPTER 1

Solving Problems by Searching



**Exercise 1: Water jugs problem**

Given two water jugs with capacities of 5 and 2 liters. Initially, the 5-liter jug is completely full, and the 2-liter jug is completely empty. The jugs don't have markings to allow measuring smaller quantities. We want to keep only 1 liter in the 2-liter jug and 4 liters in the 5-liter jug. We can use a third jug of unknown capacity but greater than 5 liters (see Figure 1.37). Propose an ideal formulation for this problem and solve it using the BFS algorithm.

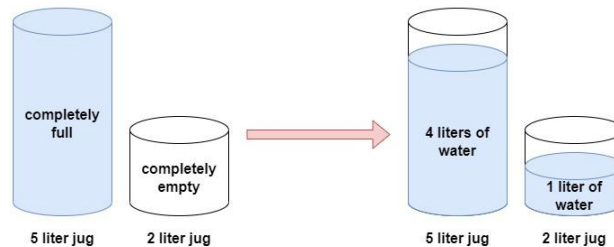


Figure 1. 1 Water jugs problem

**Solution:****1. Problem formulation**

- The state is defined by the triplet  $\langle x, y, z \rangle$ , where  $x$  represents the quantity of water in the 5-liter jug,  $y$  is the quantity of water in the 2-liter jug, and  $z$  is the quantity of water in the extra jug.
- The initial state is  $\langle 5, 0, 0 \rangle$ .
- The goal state is  $\langle 4, 1, 0 \rangle$ .
- The possible actions are:
  - (a1): Pour water from the 5-liter jug into the 2-liter jug:  $\langle x, y, z \rangle \rightarrow \langle x-d, y+d, z \rangle$ , where  $0 < d \leq 2$ .
  - (a2): Pour water from the 2-liter jug into the 5-liter jug:  $\langle x, y, z \rangle \rightarrow \langle x+d, y-d, z \rangle$ , where  $0 < d \leq 2$ .
  - (a3): Pour water from the 5-liter jug into the extra jug:  $\langle x, y, z \rangle \rightarrow \langle x-d, y, z+d \rangle$ , where  $0 < d \leq 5$ .
  - (a4): Pour water from the 2-liter jug into the extra jug:  $\langle x, y, z \rangle \rightarrow \langle x, y-d, z+d \rangle$ , where  $0 < d \leq 2$ .
  - (a5): Transfer water from the extra jug to the 5-liter jug:  $\langle x, y, z \rangle \rightarrow \langle x+d, y, z-d \rangle$ , where  $0 < d \leq 5$ .
  - (a6): Transfer water from the extra jug to the 2-liter jug:  $\langle x, y, z \rangle \rightarrow \langle x, y+d, z-d \rangle$ , where  $0 < d \leq 2$ .

**2. Problem solving with BFS**

The execution of the BFS algorithm and the search tree are shown in Table 1.6 and Figure 1.38, respectively.

The solution is: (a1)  $\rightarrow$  (a3)  $\rightarrow$  (a2)  $\rightarrow$  (a6)  $\rightarrow$  (a2)  $\rightarrow$  (a6)

The path is:  $\langle 5, 0, 0 \rangle \rightarrow \langle 3, 2, 0 \rangle \rightarrow \langle 0, 2, 3 \rangle \rightarrow \langle 2, 0, 3 \rangle \rightarrow \langle 2, 2, 1 \rangle \rightarrow \langle 4, 0, 1 \rangle \rightarrow \langle 4, 1, 0 \rangle$ .

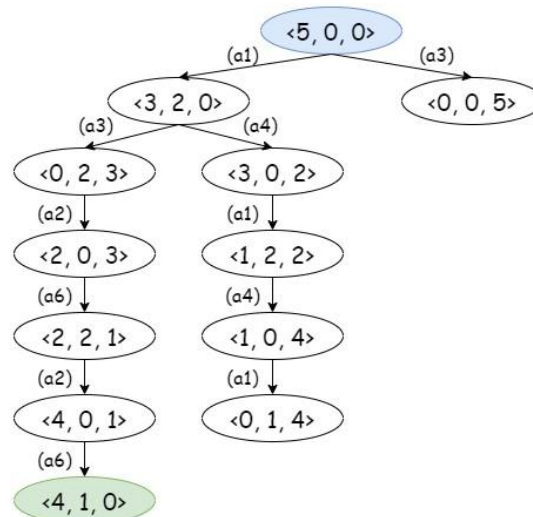


Figure 1.2 BFS search tree for the water jugs problem

Table 1.1 BFS algorithm execution for the water jugs problem

Open	Closed	Description
<5, 0, 0>		
<3, 2, 0>, <0, 0, 5>	<5, 0, 0>	
<0, 0, 5>, <0, 2, 3>, <3, 0, 2>	<5, 0, 0>, <3, 2, 0>	State <5, 0, 0> is already in Closed
<0, 2, 3>, <3, 0, 2>	<5, 0, 0>, <3, 2, 0>, <0, 0, 5>	State <5, 0, 0> is already in Closed State <0, 2, 3> is already in Open
<3, 0, 2>, <2, 0, 3>	<5, 0, 0>, <3, 2, 0>, <0, 0, 5>, <0, 2, 3>	States <0, 0, 5> and <3, 2, 0> are already in Closed
<2, 0, 3>, <1, 2, 2>	<5, 0, 0>, <3, 2, 0>, <0, 0, 5>, <0, 2, 3>, <3, 0, 2>	States <0, 0, 5>, <5, 0, 0> and <3, 2, 0> are already in Closed
<1, 2, 2>, <2, 2, 1>	<5, 0, 0>, <3, 2, 0>, <0, 0, 5>, <0, 2, 3>, <3, 0, 2>, <2, 0, 3>	States <0, 2, 3>, <0, 0, 5> and <5, 0, 0> are already in Closed
<2, 2, 1>, <1, 0, 4>	<5, 0, 0>, <3, 2, 0>, <0, 0, 5>, <0, 2, 3>, <3, 0, 2>, <2, 0, 3>, <1, 2, 2>	States <3, 0, 2>, <0, 2, 3> and <3, 2, 0> are already in Closed
<1, 0, 4>, <4, 0, 1>	<5, 0, 0>, <3, 2, 0>, <0, 0, 5>, <0, 2, 3>, <3, 0, 2>, <2, 0, 3>, <1, 2, 2>, <2, 2, 1>	States <0, 2, 3>, <2, 0, 3> and <3, 2, 0> are already in Closed
<4, 0, 1>, <0, 1, 4>	<5, 0, 0>, <3, 2, 0>, <0, 0, 5>, <0, 2, 3>, <3, 0, 2>, <2, 0, 3>, <1, 2, 2>, <2, 2, 1>, <1, 0, 4>	States <0, 0, 5>, <5, 0, 0> and <1, 2, 2> are already in Closed
<0, 1, 4>, <4, 1, 0>	<5, 0, 0>, <3, 2, 0>, <0, 0, 5>, <0, 2, 3>, <3, 0, 2>, <2, 0, 3>, <1, 2, 2>, <2, 2, 1>, <1, 0, 4>, <4, 0, 1>	States <2, 2, 1>, <0, 0, 5> and <5, 0, 0> are already in Closed Goal state <4, 1, 0> is reached

**Exercise 2: Grammar problem**

We are given the sequence *ABBEACC*. We can transform this sequence using the following rules:

1.  $AC \rightarrow E$

2.  $AB \rightarrow BC$
3.  $BB \rightarrow E$
4.  $Ex \rightarrow x$  for any  $x \in \{A, B, C, E\}$

Our objective is to produce the sequence  $E$ .

Propose a formulation for this problem and solve it using the Breadth-First Search (BFS) algorithm.

### Solution:

#### 1. Problem formulation

- The state is defined by the sequence of characters.
- The initial state is the sequence  $ABBEACC$ .
- The goal state is the sequence  $E$ .
- The possible actions are:
  - (a1):  $AC \rightarrow E$
  - (a2):  $AB \rightarrow BC$
  - (a3):  $BB \rightarrow E$
  - (a4):  $Ex \rightarrow x$  for any  $x \in \{A, B, C, E\}$

#### 2. Problem solving with BFS

The execution of the BFS algorithm and the search tree are shown in Table 1.6 and Figure 1.38, respectively.

The solution is: (a3)  $\rightarrow$  (a4)  $\rightarrow$  (a4)  $\rightarrow$  (a1)  $\rightarrow$  (a4)  $\rightarrow$  (a1)

The path is:  $ABBEACC \rightarrow AEEACC \rightarrow AEACC \rightarrow AACC \rightarrow AEC \rightarrow AC \rightarrow E$ .

Table 1. 2 BFS algorithm execution for the Grammar problem

Open	Closed	Description
<b>ABBEACC</b>		
<b>BCBEACC</b> , AEEACC, ABBACC, ABBEAC	ABBEACC	
<b>AEEACC</b> , ABBACC, ABBEAC, BCBACC, BCBEEC	ABBEACC, BCBEACC	
<b>ABBACC</b> , ABBEAC, BCBACC, BCBEEC, AEACC, AEEEC	ABBEACC, BCBEACC, AEEACC	State AEACC is already in Open
<b>ABBEAC</b> , BCBACC, BCBEEC, AEACC, AEEEC, ABBEC	ABBEACC, BCBEACC, AEEACC, ABBACC	States BCBACC and AEACC already in Open
<b>BCBACC</b> , BCBEEC, AEACC, AEEEC, ABBEC	ABBEACC, BCBEACC, AEEACC, ABBACC, ABBEAC	States BCBEEC, AEEEC, and ABBEC are already in Open
<b>BCBEEC</b> , AEACC, AEEEC, ABBEC, BCBEC	ABBEACC, BCBEACC, AEEACC, ABBACC, ABBEAC, BCBACC	
<b>AEACC</b> , AEEEC, ABBEC, BCBEC	ABBEACC, BCBEACC, AEEACC, ABBACC,	State BCBEC is already in Open

	ABBEEC, BCBACC, BCBEEC	
AEEEC, ABBEC, BCBEC, AACC, AEEC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC	
ABBECC, BCBEC, AACC, AEEC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC	State AEEC is already in Open
BCBEC, AACC, AEEC, ABBC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC, ABBEC	States BCBEC and AEEC are already in Open
AACC, AEEC, ABBC, BCBEC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC, ABBEC, BCBEC	
AEEC, ABBC, BCBEC, AEC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC, ABBEC, BCBEC, AACC	State AEC is already in Open
ABBC, BCBEC, AEC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC, ABBEC, BCBEC, AACC, AEEC	
BCBEC, AEC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC, ABBEC, BCBEC, AACC, AEEC, ABBC	States BCBEC and AEC are already in Open
AEC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC, ABBEC, BCBEC, AACC, AEEC, ABBC, BCBEC	State BCBEC has no successors
AC	ABBEACC, BCBEEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC, ABBEC, BCBEC, AACC, AEEC, ABBC,	

	BCBC	
E	ABBEACC, BCBEACC, AEEACC, ABBACC, ABBEEC, BCBACC, BCBEEC, AEACC, AEEEC, ABBEC, BCBEC, AACC, AEEC, ABBC, BCBC, AC	Goal state E is reached

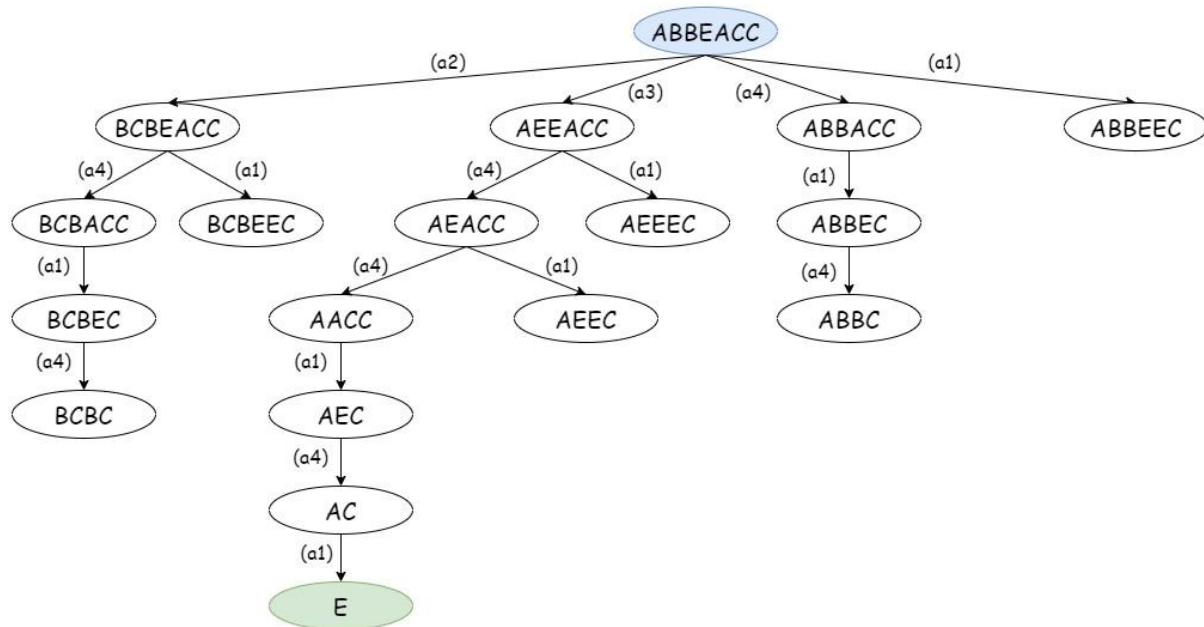


Figure 1. 3 BFS search tree for the Grammar problem

**Exercise 3: Maze Problem**

Consider a robot placed in a maze (a matrix with walls) as shown in Figure 1.39. This robot has to reach the destination marked as the *Goal* position. Find the possible path and its cost that the robot can take from the source to the destination using DFS and A\* algorithms.

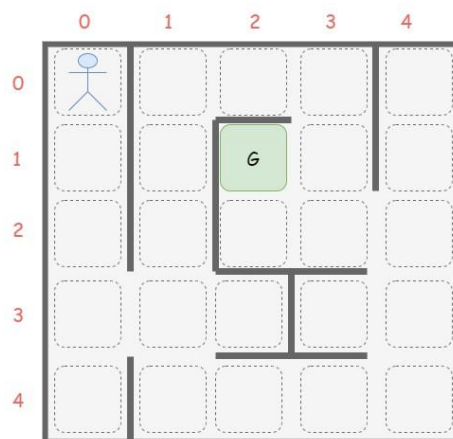


Figure 1. 4 Maze problem

**Solution:**

- The state is defined by the position of the agent as  $\langle y, x \rangle$ , where  $y$  represents the vertical coordinate,

and  $x$  represents the horizontal coordinate. Additionally, the configuration of the board, including obstacle positions and the goal position, is stored as a static attribute.

- The initial state is defined as  $\langle 0, 0 \rangle$ .
- The goal state is defined as  $\langle 1, 2 \rangle$ .
- There are four possible actions, each represented by a letter and associated with a movement direction. These actions are:

**(D)** for moving down:  $\langle y, x \rangle \rightarrow \langle y+1, x \rangle$ , with the precondition that there is no obstacle between the positions  $\langle y, x \rangle$  and  $\langle y+1, x \rangle$ .

**(R)** for moving right:  $\langle y, x \rangle \rightarrow \langle y, x+1 \rangle$ , with the precondition that there is no obstacle between the positions  $\langle y, x \rangle$  and  $\langle y, x+1 \rangle$ .

**(U)** for moving up:  $\langle y, x \rangle \rightarrow \langle y-1, x \rangle$ , with the precondition that there is no obstacle between the positions  $\langle y, x \rangle$  and  $\langle y-1, x \rangle$ .

**(L)** for moving left:  $\langle y, x \rangle \rightarrow \langle y, x-1 \rangle$ , with the precondition that there is no obstacle between the positions  $\langle y, x \rangle$  and  $\langle y, x-1 \rangle$ .

- **Application of DFS:** The execution of the DFS algorithm and the search tree are shown in Table 1.7 and Figure 1.40, respectively. The resulting path is:  $\langle 0, 0 \rangle \rightarrow \langle 1, 0 \rangle \rightarrow \langle 2, 0 \rangle \rightarrow \langle 3, 0 \rangle \rightarrow \langle 3, 1 \rangle \rightarrow \langle 4, 1 \rangle \rightarrow \langle 4, 2 \rangle \rightarrow \langle 4, 3 \rangle \rightarrow \langle 4, 4 \rangle \rightarrow \langle 3, 4 \rangle \rightarrow \langle 2, 4 \rangle \rightarrow \langle 2, 3 \rangle \rightarrow \langle 1, 3 \rangle \rightarrow \langle 1, 2 \rangle$  with a path cost of 13.

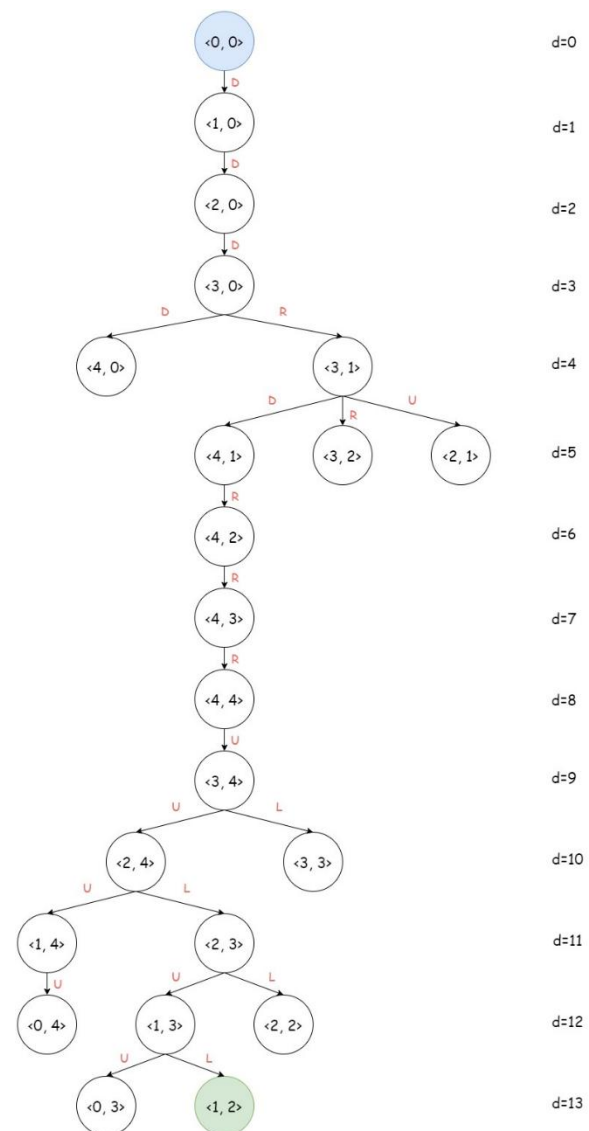


Table 1. 3 DFS algorithm execution for the maze

Figure 1. 5 DFS search tree for the maze problem

problem

Open	Closed	Description
------	--------	-------------

<0, 0>		
<1, 0>	<0, 0>	
<2, 0>	<0, 0>, <1, 0>	<0, 0> is already in Closed
<3, 0>	<0, 0>, <1, 0>, <2, 0>	<1, 0> is already in Closed
<4, 0>, <3, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>	<2, 0> is already in Closed
<3, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>	<3, 0> is already in Closed
<4, 1>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>	<3, 0> is already in Closed
<4, 2>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>	<3, 1> is already in Closed
<4, 3>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>	<4, 1> is already in Closed
<4, 4>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>, <4, 3>	<4, 2> is already in Closed
<3, 4>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>, <4, 3>, <4, 4>	<4, 3> is already in Closed
<2, 4>, <3, 3>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>, <4, 3>, <4, 4>, <3, 4>	<4, 4> is already in Closed
<1, 4>, <2, 3>, <3, 3>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>, <4, 3>, <4, 4>, <3, 4>, <2, 4>	<3, 4> is already in Closed
<0, 4>, <2, 3>, <3, 3>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>, <4, 3>, <4, 4>, <3, 4>, <2, 4>, <1, 4>	<2, 4> is already in Closed
<2, 3>, <3, 3>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>, <4, 3>, <4, 4>, <3, 4>, <2, 4>, <1, 4>, <0, 4>	<1, 4> is already in Closed
<1, 3>, <2, 2>, <3, 3>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>, <4, 3>, <4, 4>, <3, 4>, <2, 4>, <1, 4>, <0, 4>	<2, 4> is already in Closed
<0, 3>, <1, 2>, <2, 2>, <3, 3>, <3, 2>, <2, 1>	<0, 0>, <1, 0>, <2, 0>, <3, 0>, <4, 0>, <3, 1>, <4, 1>, <4, 2>, <4, 3>, <4, 4>, <3, 4>, <2, 4>, <1, 4>, <0, 4>, <1, 3>	<2, 4> is already in Closed Goal state <1, 2> is reached

- **Application of A\*:** Consider  $g(n)$  = Depth of node  $n$  and  $h(n)$  = The Manhattan distance from the agent position to the goal. The execution of the A\* algorithm and the search tree are shown in Table 1.8 and Figure 1.41, respectively. The resulting path is: <0, 0> → <1, 0> → <2, 0> → <3, 0> → <3, 1> → <2, 1> → <1, 1> → <0, 1> → <0, 2> → <0, 3> → <1, 3> → <1, 2> with a path cost of 11.

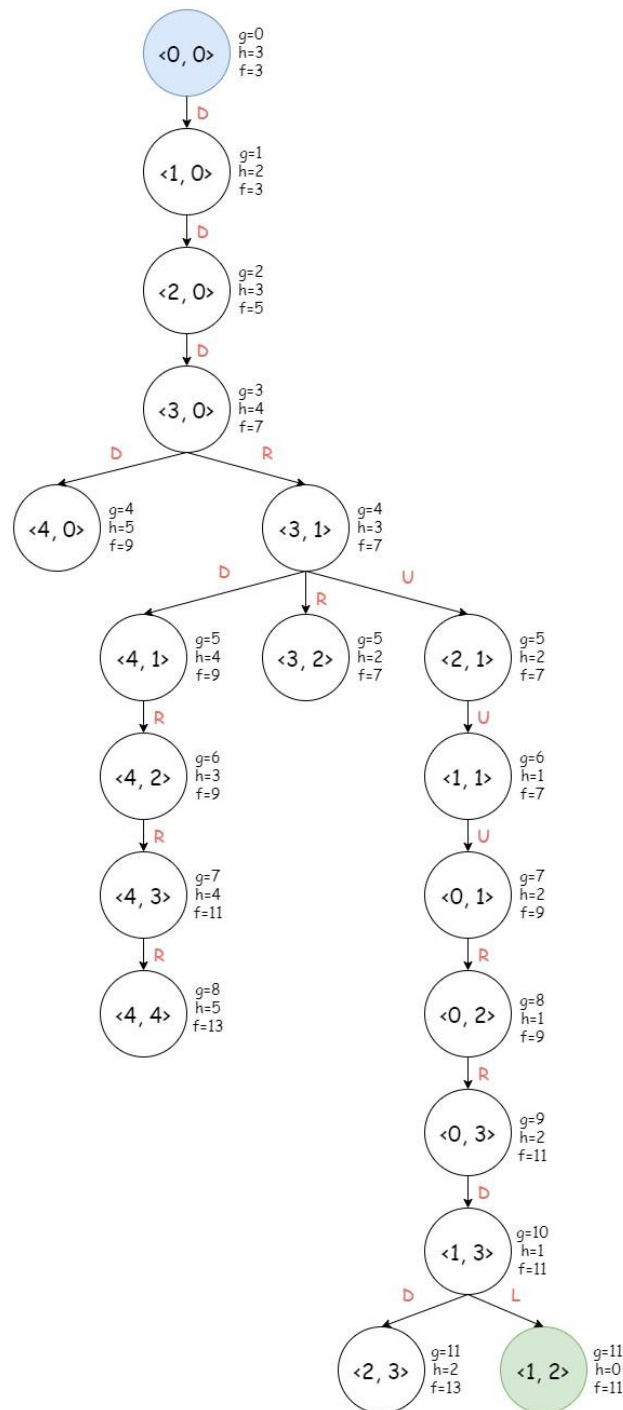


Figure 1.6 A\* search tree for the maze problem

Table 1.4 A\* algorithm execution for the maze problem

Open	Closed	Description
$\langle 0, 0 \rangle, 0, 3$		
$\langle 1, 0 \rangle, 1, 2$	$\langle 0, 0 \rangle, 0, 3$	
$\langle 2, 0 \rangle, 2, 3$	$\langle 0, 0 \rangle, 0, 3), (\langle 1, 0 \rangle, 1, 2)$	$\langle 0, 0 \rangle$ is already in Closed
$\langle 3, 0 \rangle, 3, 4$	$\langle 0, 0 \rangle, 0, 3), (\langle 1, 0 \rangle, 1, 2), (\langle 2, 0 \rangle, 2, 3)$	$\langle 1, 0 \rangle$ is already in Closed
$\langle 4, 0 \rangle, 4, 5), (\langle 3, 1 \rangle, 4, 3)$	$\langle 0, 0 \rangle, 0, 3), (\langle 1, 0 \rangle, 1, 2), (\langle 2, 0 \rangle, 2, 3), (\langle 3, 0 \rangle, 3, 4)$	$\langle 2, 0 \rangle$ is already in Closed

1

	(<0, 2>, 8, 1), (<0, 3>, 9, 2), (<1, 3>, 10, 1)	
(<4, 4>, 8, 5), (<2, 3>, 11, 2)	(<0, 0>, 0, 3), (<1, 0>, 1, 2), (<2, 0>, 2, 3), (<3, 0>, 3, 4), (<3, 1>, 4, 3), (<3, 2>, 5, 2), (<2, 1>, 5, 2), (<1, 1>, 6, 1), (<4, 0>, 4, 5), (<4, 1>, 5, 4), (<0, 1>, 7, 2), (<4, 2>, 6, 3), (<0, 2>, 8, 1), (<0, 3>, 9, 2), (<1, 3>, 10, 1)	Goal state is reached: (<1, 2>, 11, 0)

#### Exercise 4: Battlefield Problem

1. In a strategy game, a unit needs to navigate a battlefield represented as a 2D grid, where each cell has a terrain type that influences the movement cost. The objective is to move the unit from its starting position (0, 0) to a target position (4, 4) with the minimum total cost, while avoiding enemy zones and impassable terrain. The terrain types are:

- P (Plain): Normal cost of 1.
- F (Forest): Higher cost (cost of 3).
- M (Mountain): Impassable terrain.
- E (Enemy Zone): Avoid if possible, extremely high cost (cost of 10).

Solve this game using the A\* algorithm, ensuring that the path avoids enemy zones and impassable terrain.

	0	1	2	3	4
0	P	P	F	P	P
1	P	F	E	P	P
2	P	P	F	P	P
3	M	M	F	P	P
4	P	P	P	P	P

#### Solution:

State: (x, y) => Node [(x, y), g, h] (x: vertical coordinate, y: horizontal coordinate)

Initial state: (0, 0) => Initial node [(0, 0), 0, 8]

Goal state: (4, 4)

Actions: R (right), L (left), D (down), U (up)

**Path Cost (g):** The path cost  $g(n)$  is the accumulated movement cost from the **start state** to the current state.

For example:

- Moving through **Plain (P)** cells:  $g$  increases by 1 per step.
- Moving through **Forest (F)** cells:  $g$  increases by 3 per step.
- Moving through **Enemy Zone (E)** cells:  $g$  increases by 10 per step.

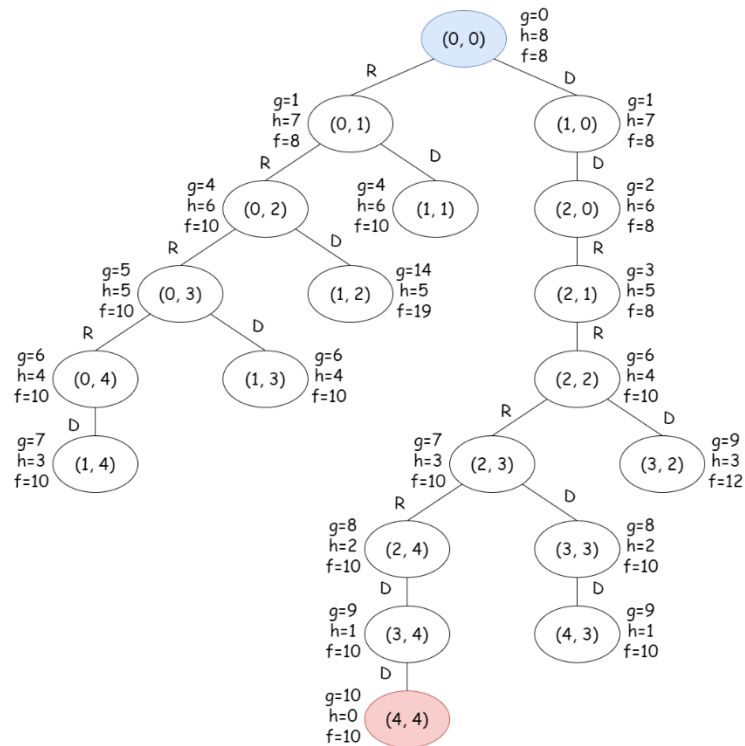
**Heuristic Function (h):** The heuristic estimates the cost from the current state to the goal state. We use the Manhattan Distance.

**Objective:** Find a path from (0, 0) to (4, 4) that minimizes the total cost  $f(n)$ , where:  $f(n)=g(n)+h(n)$ .

OPEN	CLOSED
[(0,0),0,8]	
[(0,1),1,7], [(1,0),1,7]	[(0,0),0,8]

[(1,0),1,7], [(0,2),4,6], [(1,1),4,6]	[(0,0),0,8], [(0,1),1,7]
[(0,2),4,6], [(1,1),4,6], [(2,0),2,6]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7]
[(0,2),4,6], [(1,1),4,6], [(2,1),3,5]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6]
[(0,2),4,6], [(1,1),4,6], [(2,2),6,4]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5]
[(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(1,2),14,5],	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6]
[(2,2),6,4], [(0,3),5,5], [(1,2),14,5]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6]
[(0,3),5,5], [(1,2),14,5], [(2,3),7,3], [(3,2),9,3]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4]
[(1,2),14,5], [(2,3),7,3], [(3,2),9,3], [(0,4),6,4], [(1,3),6,4]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5]
[(1,2),14,5], [(3,2),9,3], [(0,4),6,4], [(1,3),6,4], [(2,4),8,2], [(3,3),8,2]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5], [(2,3),7,3]
[(1,2),14,5], [(3,2),9,3], [(1,3),6,4], [(2,4),8,2], [(3,3),8,2], [(1,4),7,3]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5], [(2,3),7,3], [(0,4),6,4]
[(1,2),14,5], [(3,2),9,3], [(2,4),8,2], [(3,3),8,2], [(1,4),7,3]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5], [(2,3),7,3], [(0,4),6,4], [(1,3),6,4]
[(1,2),14,5], [(3,2),9,3], [(3,3),8,2], [(1,4),7,3], [(3,4),9,1]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5], [(2,3),7,3], [(0,4),6,4], [(1,3),6,4], [(2,4),8,2]
[(1,2),14,5], [(3,2),9,3], [(1,4),7,3], [(3,4),9,1], [(4,3),9,1]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5], [(2,3),7,3], [(0,4),6,4], [(1,3),6,4], [(2,4),8,2], [(3,3),8,2]
[(1,2),14,5], [(3,2),9,3], [(3,4),9,1], [(4,3),9,1]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5], [(2,3),7,3], [(0,4),6,4], [(1,3),6,4], [(2,4),8,2], [(3,3),8,2], [(1,4),7,3]
[(1,2),14,5], [(3,2),9,3], [(4,3),9,1], [(4,4),10,0]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5], [(2,3),7,3], [(0,4),6,4], [(1,3),6,4], [(2,4),8,2], [(3,3),8,2], [(1,4),7,3], [(3,4),9,1]
[(1,2),14,5], [(3,2),9,3], [(4,4),10,0]	[(0,0),0,8], [(0,1),1,7], [(1,0),1,7], [(2,0),2,6], [(2,1),3,5], [(0,2),4,6], [(1,1),4,6], [(2,2),6,4], [(0,3),5,5], [(2,1),5,5], [(2,3),7,3], [(0,4),6,4],

[(1,3),6,4], [(2,4),8,2], [(3,3),8,2], [(1,4),7,3],  
[(3,4),9,1], [(4,3),9,1]



Path:

	0	1	2	3	4
0	P	P	F	P	P
1	P	F	E	P	P
2	P	P	F	P	P
3	M	M	F	P	P
4	P	P	P	P	P

Cost: 10

Solution: D -> D -> D -> R -> R -> R -> R -> D -> D

#### Exercise 4: 8-puzzle problem

Given an initial state of an 8-puzzle problem and the goal state to be reached in Figure 1.42. Find the most cost-effective path to reach the goal state from the initial state using A\* algorithm.

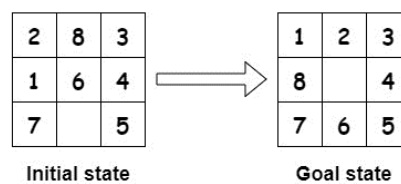


Figure 1. 7 8-puzzle problem

**Solution:**

- The state is defined by the configuration of the 8-puzzle, which includes the locations of each of the eight tiles and the blank space within one of the nine squares.

2	8	3
1	6	4
7		5

- The initial state is

1	2	3
8		4
7	6	5

- The goal state is

- There four possible actions that represent movements of the blank space:

**(L)** for moving the blank space left, with the precondition that the blank space is not at the left limit of the puzzle.

**(U)** for moving the blank space up, with the precondition that the blank space is not at the top limit of the puzzle.

**(R)** for moving the blank space right, with the precondition that the blank space is not at the right limit of the puzzle.

**(D)** for moving the blank space down, with the precondition that the blank space is not at the down limit of the puzzle.

- **Application of A\*:** Consider  $g(n)$  = Depth of node and  $h(n)$  = Number of misplaced tiles. The execution of the A\* algorithm and the search tree are shown in Table 1.9 and Figure 1.43,

respectively. The resulting path is:

2	8	3
1	6	4
7		5

→

2	8	3
1		4
7	6	5

→

2		3
1	8	4
7	6	5

→

	2	3
1	8	4
7	6	5

→

1	2	3
	8	4
7	6	5

→

1	2	3
8		4
7	6	5

with a path cost of 5.

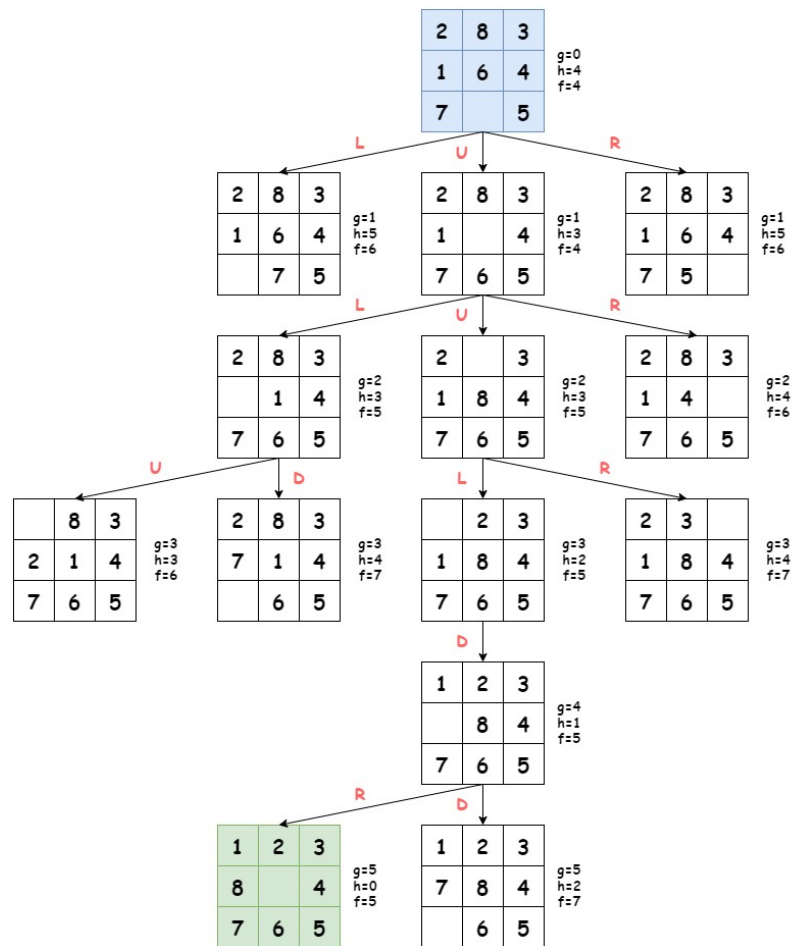


Figure 1.8 A\* search tree for the 8-puzzle problem

Table 1.5 A\* algorithm execution for the 8-puzzle problem

Open	Closed	Description
<div>2 8 3</div> <div>1 6 4</div> <div>7 5 5</div>		
<div>2 8 3</div> <div>1 4 1 6 4 1 6 4</div> <div>7 6 5 7 5 7 5</div>	<div>2 8 3</div> <div>1 6 4</div> <div>7 5</div>	
<div>2 8 3</div> <div>1 4 1 8 4 1 6 4 1 6 4 1 4</div> <div>7 6 5 7 6 5 7 5 7 5 7 6 5</div>	<div>2 8 3 2 8 3</div> <div>1 6 4 1 4</div> <div>7 5 7 6 5</div>	<div>2 8 3</div> <div>1 6 4</div> <div>7 5</div> is already in Closed
<div>2 8 3</div> <div>1 8 4 1 6 4 1 6 4 1 4 2 1 4</div> <div>7 6 5 7 5 7 5 7 6 5 7 6 5</div> <div>2 8 3</div> <div>7 1 4</div> <div>6 5</div>	<div>2 8 3 2 8 3 2 8 3</div> <div>1 6 4 1 4 1 4</div> <div>7 5 7 6 5 7 6 5</div>	<div>2 8 3</div> <div>1 4</div> <div>7 6 5</div> is already in Closed
<div>2 3</div> <div>1 8 4 1 6 4 1 6 4 1 4 2 1 4</div> <div>7 6 5 7 5 7 5 7 6 5 7 6 5</div> <div>2 8 3 2 3</div> <div>7 1 4 1 8 4</div> <div>6 5 7 6 5</div>	<div>2 8 3 2 8 3 2 8 3 2 3</div> <div>1 6 4 1 4 1 4 1 8 4</div> <div>7 5 7 6 5 7 6 5 7 6 5</div>	<div>2 8 3</div> <div>1 4</div> <div>7 6 5</div> is already in Closed
<div>1 2 3</div> <div>1 8 4 1 6 4 1 6 4 1 4 2 1 4</div> <div>7 6 5 7 5 7 5 7 6 5 7 6 5</div>	<div>2 8 3 2 8 3 2 8 3 2 3</div> <div>1 6 4 1 4 1 4 1 8 4 1 8 4</div> <div>7 5 7 6 5 7 6 5 7 6 5 7 6 5</div>	<div>2 3</div> <div>1 8 4</div> <div>7 6 5</div> is already in Closed

<table><tr><td>2</td><td>8</td><td>3</td><td>2</td><td>3</td></tr><tr><td>7</td><td>1</td><td>4</td><td>1</td><td>8</td><td>4</td></tr><tr><td></td><td>6</td><td>5</td><td>7</td><td>6</td><td>5</td></tr></table>	2	8	3	2	3	7	1	4	1	8	4		6	5	7	6	5																																																																																																																						
2	8	3	2	3																																																																																																																																			
7	1	4	1	8	4																																																																																																																																		
	6	5	7	6	5																																																																																																																																		
<table><tr><td>1</td><td>2</td><td>3</td><td>2</td><td>8</td><td>3</td><td>2</td><td>8</td><td>3</td><td>2</td><td>8</td><td>3</td><td></td><td>8</td><td>3</td></tr><tr><td>8</td><td></td><td>4</td><td>1</td><td>6</td><td>4</td><td>1</td><td>6</td><td>4</td><td>1</td><td>4</td><td></td><td>2</td><td>1</td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td><td></td><td>7</td><td>5</td><td>7</td><td>5</td><td></td><td>7</td><td>6</td><td>5</td><td>7</td><td>6</td><td>5</td></tr></table> <table><tr><td>2</td><td>8</td><td>3</td><td>2</td><td>3</td><td>1</td><td>2</td><td>3</td></tr><tr><td>7</td><td>1</td><td>4</td><td>1</td><td>8</td><td>4</td><td>7</td><td>8</td><td>4</td></tr><tr><td></td><td>6</td><td>5</td><td>7</td><td>6</td><td>5</td><td></td><td>6</td><td>5</td></tr></table>	1	2	3	2	8	3	2	8	3	2	8	3		8	3	8		4	1	6	4	1	6	4	1	4		2	1	4	7	6	5		7	5	7	5		7	6	5	7	6	5	2	8	3	2	3	1	2	3	7	1	4	1	8	4	7	8	4		6	5	7	6	5		6	5	<table><tr><td>2</td><td>8</td><td>3</td><td>2</td><td>8</td><td>3</td><td>2</td><td>8</td><td>3</td><td>2</td><td></td><td>3</td><td></td><td>2</td><td>3</td></tr><tr><td>1</td><td>6</td><td>4</td><td>1</td><td></td><td>4</td><td>1</td><td>4</td><td>1</td><td>8</td><td>4</td><td>1</td><td>8</td><td>4</td></tr><tr><td>7</td><td></td><td>5</td><td>7</td><td>6</td><td>5</td><td>7</td><td>6</td><td>5</td><td>7</td><td>6</td><td>5</td><td>7</td><td>6</td><td>5</td></tr></table> <table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>8</td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table>	2	8	3	2	8	3	2	8	3	2		3		2	3	1	6	4	1		4	1	4	1	8	4	1	8	4	7		5	7	6	5	7	6	5	7	6	5	7	6	5	1	2	3		8	4	7	6	5	<table><tr><td></td><td>2</td><td>3</td></tr><tr><td>1</td><td>8</td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table> <p>is already in Closed</p>		2	3	1	8	4	7	6	5
1	2	3	2	8	3	2	8	3	2	8	3		8	3																																																																																																																									
8		4	1	6	4	1	6	4	1	4		2	1	4																																																																																																																									
7	6	5		7	5	7	5		7	6	5	7	6	5																																																																																																																									
2	8	3	2	3	1	2	3																																																																																																																																
7	1	4	1	8	4	7	8	4																																																																																																																															
	6	5	7	6	5		6	5																																																																																																																															
2	8	3	2	8	3	2	8	3	2		3		2	3																																																																																																																									
1	6	4	1		4	1	4	1	8	4	1	8	4																																																																																																																										
7		5	7	6	5	7	6	5	7	6	5	7	6	5																																																																																																																									
1	2	3																																																																																																																																					
	8	4																																																																																																																																					
7	6	5																																																																																																																																					
	2	3																																																																																																																																					
1	8	4																																																																																																																																					
7	6	5																																																																																																																																					