

# Lab Assignment 1

**Subject:** Artificial Intelligence

**Guided by:** Dr. Anuradha Yenikar

**Student name:** Neha Sawant

---

**Experiment Name:** Implement DFS and BFS for the 8-puzzle problem

## Objective:

- To understand and implement the Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms for solving the 8-puzzle problem.

## Problem Statement:

The 8-puzzle problem involves arranging tiles numbered 1 to 8 on a 3x3 grid in such a way that they are in numerical order, with the blank space (denoted by 0) in the bottom-right corner. You need to solve this puzzle using BFS and DFS algorithms.

## Algorithm:

### 1. BFS (Breadth-First Search):

- Explore all nodes level by level.
- Ensure all possible configurations are generated and checked until the goal is reached.

### 2. DFS (Depth-First Search):

- Explore each branch as deeply as possible.
- Backtrack when no further moves are possible.

## Code:

```

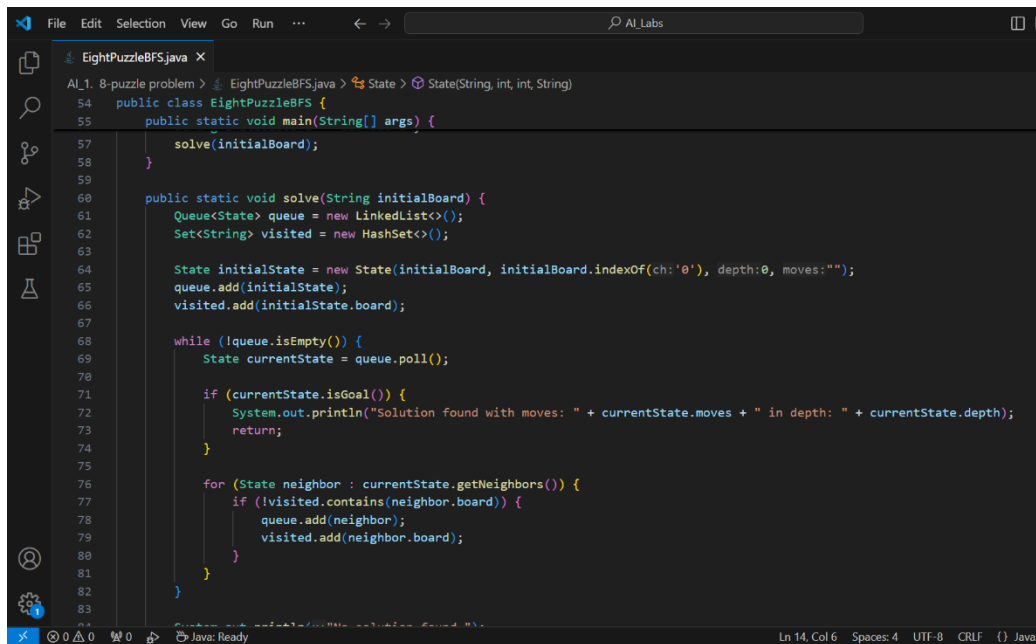
File Edit Selection View Go Run ... AI_L
EightPuzzleBFS.java X
AI_1: 8-puzzle problem > EightPuzzleBFS.java > State > State(String, int, int, String)
1  import java.util.*;
2
3  class State {
4      String board;
5      int zeroIndex;
6      int depth;
7      String moves;
8
9      State(String board, int zeroIndex, int depth, String moves) {
10         this.board = board;
11         this.zeroIndex = zeroIndex;
12         this.depth = depth;
13         this.moves = moves;
14     }
15
16
17     boolean isGoal() {
18         return board.equals(anObject:"123456780");
19     }
20
21
22     List<State> getNeighbors() {
23         List<State> neighbors = new ArrayList<>();
24         int[][] directions = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};
25         int row = zeroIndex / 3;
26         int col = zeroIndex % 3;
27
28         for (int[] dir : directions) {
29             int newRow = row + dir[0];
30             int newCol = col + dir[1];

```

```

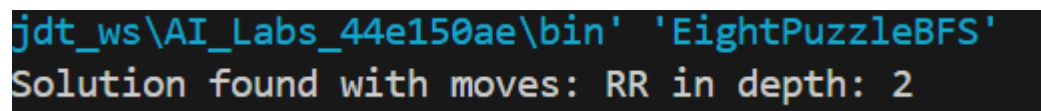
File Edit Selection View Go Run ... AI_Labs
EightPuzzleBFS.java X
AI_1: 8-puzzle problem > EightPuzzleBFS.java > State > State(String, int, int, String)
22     List<State> getNeighbors() {
31         if (newRow >= 0 && newRow < 3 && newCol >= 0 && newCol < 3) {
32             int newIndex = newRow * 3 + newCol;
33             // Swap zero with the target tile
34             char[] newBoard = board.toCharArray();
35             newBoard[zeroIndex] = newBoard[newIndex];
36             newBoard[newIndex] = '0';
37             neighbors.add(new State(new String(newBoard), newIndex, depth + 1, moves + getMove(zeroIndex, newIndex)));
38         }
39     }
40
41     return neighbors;
42 }
43
44
45 private String getMove(int oldIndex, int newIndex) {
46     if (newIndex == oldIndex + 3) return "D";
47     if (newIndex == oldIndex - 3) return "U";
48     if (newIndex == oldIndex + 1) return "R";
49     if (newIndex == oldIndex - 1) return "L";
50     return "";
51 }
52 }
53
54 public class EightPuzzleBFS {
55     Run | Debug
56     public static void main(String[] args) {
57         String initialBoard = "123456078";
58         solve(initialBoard);
59     }

```



```
File Edit Selection View Go Run ... AI_Labs
EightPuzzleBFS.java
AI_1. 8-puzzle problem > EightPuzzleBFS.java > State > State(String, int, int, String)
54 public class EightPuzzleBFS {
55     public static void main(String[] args) {
56         solve(initialBoard);
57     }
58
59     public static void solve(String initialBoard) {
60         Queue<State> queue = new LinkedList<>();
61         Set<String> visited = new HashSet<>();
62
63         State initialState = new State(initialBoard, initialBoard.indexOf(ch:'0'), depth:0, moves:"");
64         queue.add(initialState);
65         visited.add(initialState.board);
66
67         while (!queue.isEmpty()) {
68             State currentState = queue.poll();
69
70             if (currentState.isGoal()) {
71                 System.out.println("Solution found with moves: " + currentState.moves + " in depth: " + currentState.depth);
72                 return;
73             }
74
75             for (State neighbor : currentState.getNeighbors()) {
76                 if (!visited.contains(neighbor.board)) {
77                     queue.add(neighbor);
78                     visited.add(neighbor.board);
79                 }
80             }
81         }
82     }
83 }
```

### Output Example:



```
jdt_ws\AI_Labs_44e150ae\bin' 'EightPuzzleBFS'
Solution found with moves: RR in depth: 2
```

### Conclusion:

- BFS explores nodes level by level, ensuring the shortest path is found.
- DFS, when implemented, would explore a path until no further moves are possible before backtracking.