Temple Trap Puzzle: Problem Statement and Test Configurations

1 Introduction

The *Temple Trap* puzzle is a deterministic single-agent planning problem involving layered environments, constrained connectivity, and movable tiles. The objective is to guide a pawn from an initial configuration to an exit cell using valid pawn movements and tile slides while minimizing the total cost.

2 Board Configuration

The puzzle is played on a 3×3 grid indexed in row-major order:

$$\begin{array}{ccc} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{array}$$

The grid contains exactly eight tiles and one empty cell.

3 Tile Orientation and Sides

Each tile has a fixed orientation identified by a unique symbol in the bottom-left corner. Tile edges are labeled clockwise:

- Side I: top
- Side II: right
- Side III: bottom
- Side IV: left

Each tile specifies openings separately for the top and ground layers.

4 Tile Definitions

Each tile is characterized by:

- Open sides on the top layer
- Open sides on the ground layer
- Presence of a hole
- Presence of stairs

Top-Only Corridor Tiles

- **A**: Top opens on Sides I, II; no ground openings; no hole; no stairs.
- **B**: Top opens on Sides I, II; no ground openings; no hole; no stairs.
- **C**: Top opens on Sides II, IV; no ground openings; no hole; no stairs.

Stair Tiles with Holes

- **D**: Top opens on Side IV; ground opens on Side II; hole present; stairs present.
- **E**: Top opens on Side IV; ground opens on Side II; hole present; stairs present.

Bottom-Only Hole Tiles

- **F**: Ground opens on Sides I, II; hole present; no stairs.
- **G**: Ground opens on Sides I, II; hole present; no stairs.
- **H**: Ground opens on Sides I, II; hole present; no stairs.

5 Movement and Sliding Rules

The pawn may move between adjacent tiles if the corresponding sides are open on the current layer. Vertical movement between layers is possible only via stair tiles. The pawn cannot enter the empty cell.

A tile may slide into the empty cell only if the pawn is not currently on that tile. Each pawn movement and tile slide has a cost of 1.

6 Exit and Goal Condition

The exit is located at cell 0 on the left boundary of the board. The puzzle is solved if the pawn can reach cell 0 and the tile at that cell has its left edge open.

7 Test Cases

This section lists **all test cases** used to validate the solver. Each test case specifies the initial board configuration, tile rotations, pawn position, and pawn floor.

7.1 Easy Instances

```
initial_board = ['C','D','G','B',' ','H','A','E','F']
rotations = [0,0,2,3,0,1,0,0,2]
pawn_pos = 8
pawn_floor = 'ground'
```

```
initial_board = ['A','E','G','B',' ','D','H','C','F']
rotations = [2,1,0,0,0,0,0,2,2]
pawn_pos = 1
pawn_floor = 'top'
```

```
initial_board = ['G','E','B','D','H','F','A',' ','C']
rotations = [2,3,1,0,1,0,0,0,3]
pawn_pos = 1
pawn_floor = 'ground'
```

```
initial_board = ['B','D','H','E','G','F',' ','A','C']
rotations = [3,0,2,3,0,1,0,0,0]
pawn_pos = 3
pawn_floor = 'ground'
```

7.2 Medium Instances

```
initial_board = ['G','F','E','A','B','C','H',' ','D']
rotations = [2,3,1,2,3,3,0,0,3]
pawn_pos = 6
pawn_floor = 'ground'
```

```
initial_board = ['C','E','G','F',' ','D','H','B','A']
rotations = [0,0,3,0,0,2,0,1,3]
pawn_pos = 5
pawn_floor = 'top'
```

```
initial_board = ['C','A','B','D','E','G',' ','F','H']
rotations = [3,0,2,2,0,2,0,2,3]
pawn_pos = 3
pawn_floor = 'top'
```

```
initial_board = ['G','H','C','B',' ','D','A','E','F']
rotations = [2,0,3,3,0,0,0,0,2]
pawn_pos = 7
pawn_floor = 'top'
```

7.3 Hard Instances

```
initial_board = ['D','B','C','G','F','A','H','E',' ']
rotations = [3,2,3,0,2,1,1,1,0]
pawn_pos = 0
pawn_floor = 'top'
```

```
initial_board = ['B','A','D','C','F','G',' ','H','E']
rotations = [2,0,3,0,2,3,0,1,1]
pawn_pos = 4
pawn_floor = 'ground'
```

```
initial_board = ['C','A',' ','B','H','D','E','G','F']
rotations = [3,2,0,0,3,2,0,0,2]
pawn_pos = 5
pawn_floor = 'ground'
```

```
initial_board = ['B','D','F','A','E','G','H','C',' ']
rotations = [3,0,2,0,0,2,0,3,0]
pawn_pos = 5
pawn_floor = 'ground'
```

```
initial_board = ['E','B','G',' ','C','H','F','A','D']
rotations = [2,2,2,0,3,2,2,0,0]
pawn_pos = 8
pawn_floor = 'top'
```

```
initial_board = ['F','G','B',' ','H','E','D','C','A']
rotations = [3,2,2,0,0,2,2,0,1]
pawn_pos = 0
pawn_floor = 'ground'
```

8 Objective

The objective is to compute a minimum-cost sequence of valid pawn movements and tile slides that allows the pawn to exit the board for each test configuration.