**8-Puzzle Implementation Analysis Report**

1. **Implementation of Possible Tile Moves:**

In the 8-puzzle implantation, the method for returning possible tile moves given the empty space is in the ‘possibleMoves()’ method of the ‘EightPuzzleState’ class. The approach is as follows:

1. The method first calculates the row and column of the empty space in the grid using integer division and modulo operations.
2. It then checks for up to four possible moves – up, down, left, right based on the empty space’s position.
3. For each valid move, the method:

* Creates a clone of the current state
* Swaps the empty space with the adjacent tile
* Updates the empty space position in the new state
* Adds the new state to the list of possible moves.

The implementation ensures only the valid moves are considered.

1. **Heuristic Implementation for A\* Search:**

The implementation uses the Manhattan Distance heuristic for A\* search. This heuristic calculates the sum of the horizontal and vertical distances that each tile is away from its goal position. The implementation:

1. Iterates through each position in the current board configuration
2. For each non-empty tile, it determines:

* Its current position
* Its target position in the goal state

1. Calculates the Manhattan distance for each tile as the sum of absolute difference between current and target row/column positions.
2. Sums these distances for all tiles to get the heuristic value
3. **Uniform Cost Search:**

When solving the puzzle with initial state and goal state using Unform cost search:

* + Number of moves required: 30
  + Execution time: ~336 ms
  + Nodes expanded: 181,272
  + Nodes unexpanded: 417

1. **A\* Search:**

When solving the same puzzle with A\* Search using the Manhattan Distance heuristic:

* + Number of moves required: 30
  + Execution time: ~21 ms
  + Nodes expanded: 9452
  + Nodes unexpanded: 5103