# Part3. Srinivas Bhatt. Registration : 100455111

3.1. Modifications for Handling Blocked Tiles

To efficiently handle blocked tiles in the n-tile puzzle using the IDA\* algorithm, several modifications and considerations are necessary. These changes mainly involve adjusting the state representation, successor generation, and heuristic calculation to account for the presence of obstacles.

State Representation:

The state representation must be extended to include information about the positions of blocked tiles. This could be a part of the state tuple or a separate data structure accessible during the search.

Successor Generation:

The generate\_successors method must be adapted to ensure that moves leading to or through blocked tiles are not considered valid. This involves checking the destination of a potential move against the list or map of blocked tiles and filtering out any moves that would result in a collision with an obstacle.

Heuristic Calculation:

The heuristic function (e.g., Manhattan distance) needs to be adjusted to account for blocked tiles. Since direct paths might be obstructed, the heuristic should ideally reflect the additional distance required to navigate around obstacles. This can significantly increase the complexity of the heuristic calculation, possibly requiring a precomputed map of distances that consider obstacles or an on-the-fly calculation that simulates pathfinding around the blocked tiles.

Ensure the heuristic remains admissible (never overestimates the cost to reach the goal) even in the presence of obstacles. This might mean using a simpler heuristic that underestimates the complexity added by obstacles but is easier to compute and guarantees admissibility.

Pruning and Cycle Detection:

Implement enhanced pruning strategies to avoid exploring paths that are unlikely to lead to the goal due to blocked tiles. This could involve detecting dead-end paths early and backtracking.

Maintain a mechanism for cycle detection and avoidance, especially important in puzzles with obstacles where the path to the goal might require revisiting previously explored rows or columns.

3.2. Ensuring an Optimal Solution

The IDA\* algorithm can still find an optimal solution while navigating around blocked tiles by adhering to the following principles:

Admissible Heuristic:

The key to ensuring optimality in IDA\* (or any A\*-based algorithm) is the use of an admissible heuristic, one that never overestimates the true cost to reach the goal from any given state. Even with obstacles, as long as the heuristic respects this condition, IDA\* is guaranteed to find the least-cost path to the goal.

Depth-First Search with Thresholds:

IDA\* combines depth-first search's space efficiency with the heuristic-guided exploration of A\*. By iteratively deepening the search and increasing the cost threshold based on the heuristic, the algorithm efficiently explores the most promising paths first, ensuring that the first solution found is the optimal one.

Iterative Deepening:

The iterative deepening approach ensures that all shorter paths are explored before longer ones. This systematic exploration guarantees that if a path to the goal exists that avoids obstacles, IDA\* will find the shortest such path.

Cost Accounting:

By accurately accounting for the cost of moves (including the implicit cost of detouring around obstacles), and comparing it against the heuristic-driven threshold, IDA\* ensures that it explores all viable paths without exceeding the minimal cost found in previous iterations.

In summary, by carefully adapting the heuristic to account for obstacles while maintaining its admissibility, and by methodically exploring paths within the bounds of increasingly accurate cost estimates, IDA\* can efficiently solve the n-tile puzzle with blocked tiles, ensuring optimality in the process.