# Part3. Srinivas Bhatt. Registration: 100455111

**Part 3.1**

To modify the IDA\* algorithm for solving an n-tile problem with blocked tiles, focus on adapting the state representation, successor generation, and heuristic function:

1. **State Representation**: Include the positions of blocked tiles in the puzzle's state to ensure the algorithm recognizes obstacles. This could involve including a list or set of coordinates for blocked tiles within each state representation.
2. **Successor Generation**: Modify **generate\_successors** to prevent generating states that move the blank tile into blocked positions, effectively navigating around obstacles. This means adding a check to verify that a tile is not blocked before allowing a swap between the blank tile and a numbered tile.
3. **Heuristic Adjustment**: Develop a complex heuristic (e.g., Manhattan distance) that can dynamically calculate paths around obstacles, ensuring it remains admissible by not overestimating the cost of reaching the goal state while considering detours around obstacles.
4. **Pathfinding with Obstacles**: Techniques from graph theory, such as A\* or Dijkstra's algorithm, could be used to find the shortest path around obstacles for heuristic estimation.

**Part 3.2**

To ensure the algorithm finds an optimal solution around blocked tiles:

* **Admissible Heuristic**: Maintain an admissible heuristic that accurately estimates costs without overestimation, even with obstacles, to ensure all potentially optimal paths are explored.
* **Depth-First Search with Iterative Deepening**: Use IDA\*'s iterative deepening mechanism, which, combined with a valid heuristic and move generation that considers obstacles, ensures the exploration of efficient paths to the goal.

IDA\* combines depth-first search's space efficiency with a cost threshold that increases iteratively, ensuring that it explores paths in order of increasing cost

* **Cost Function and Move Validation**: Ensure moves into blocked tiles are not allowed and that the cost function accurately reflects the cost of moves, allowing the algorithm to assess and choose the most cost-effective path to the goal.

By implementing these adjustments, IDA\* can effectively solve puzzles with blocked tiles, navigating around obstacles to find the optimal path to the goal.