Lab-3

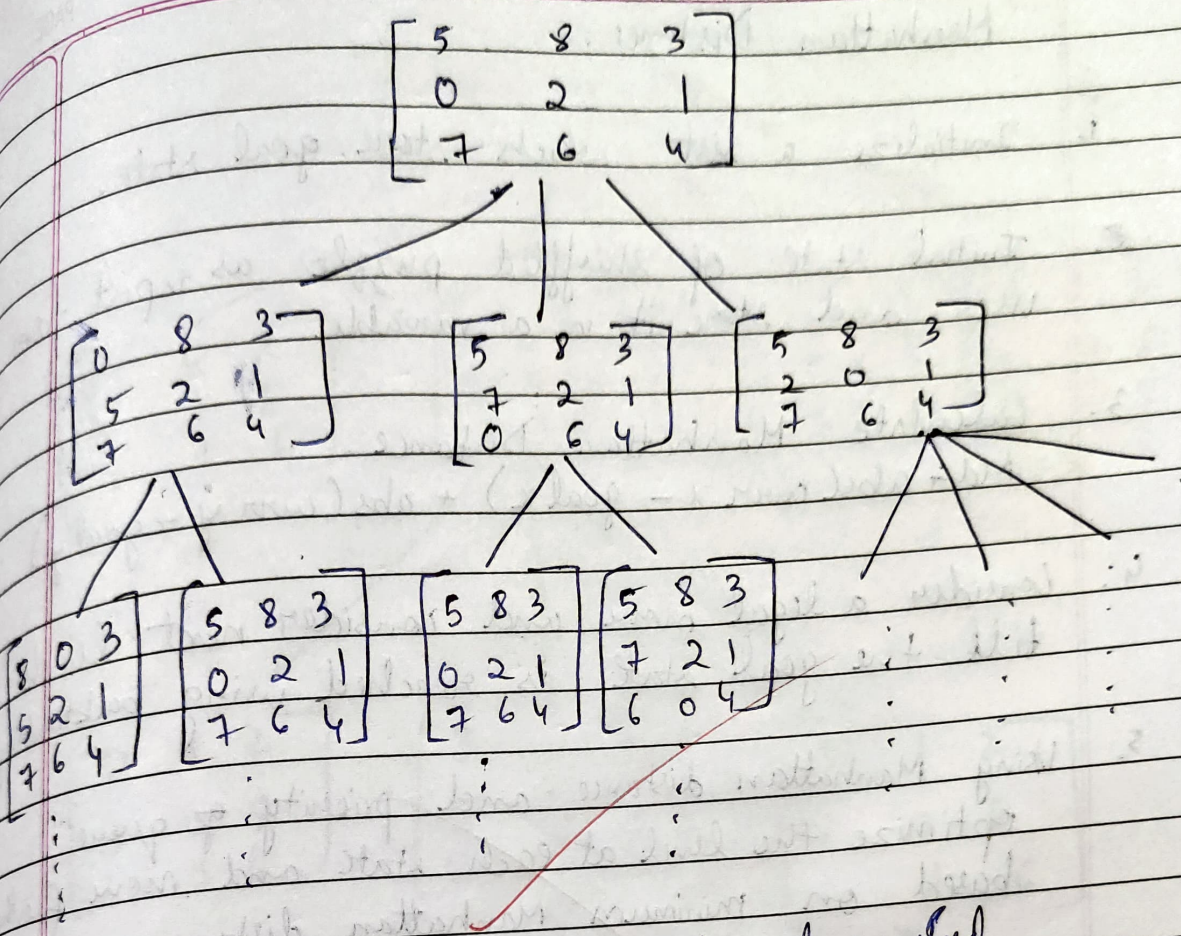
DATE:

PAGE:

Solve the 8-puzzle problem using DFS method:

1. Initialize the start and goal states respectively.
2. Find the blank tile : locate the position of the blank tile (represented by 0) in the current state
3. Check if the current state is goal state by comparing the 2.
   If they match, the puzzle is solved.
4. Mark the current state as visited by storing the current state to avoid revisiting it and causing infinite loop
5. Explore possible moves by trying to move the blank tile in 4 possible directions (right, down, left and right)
   For each valid move, create a new puzzle state by swapping the blank tile with its neighboring tile
6.
7. Recursively explore the new states by applying DFS method to explore further moves.
8. Backtrack if no solution is found from current state and try a different move.
9. End when the goal state is reached or all possibilities are exhausted.
   Return successfull if goal state is reached
   return failure if no solution is found.

DATE: PAGE:



All possible cases is checked and verified

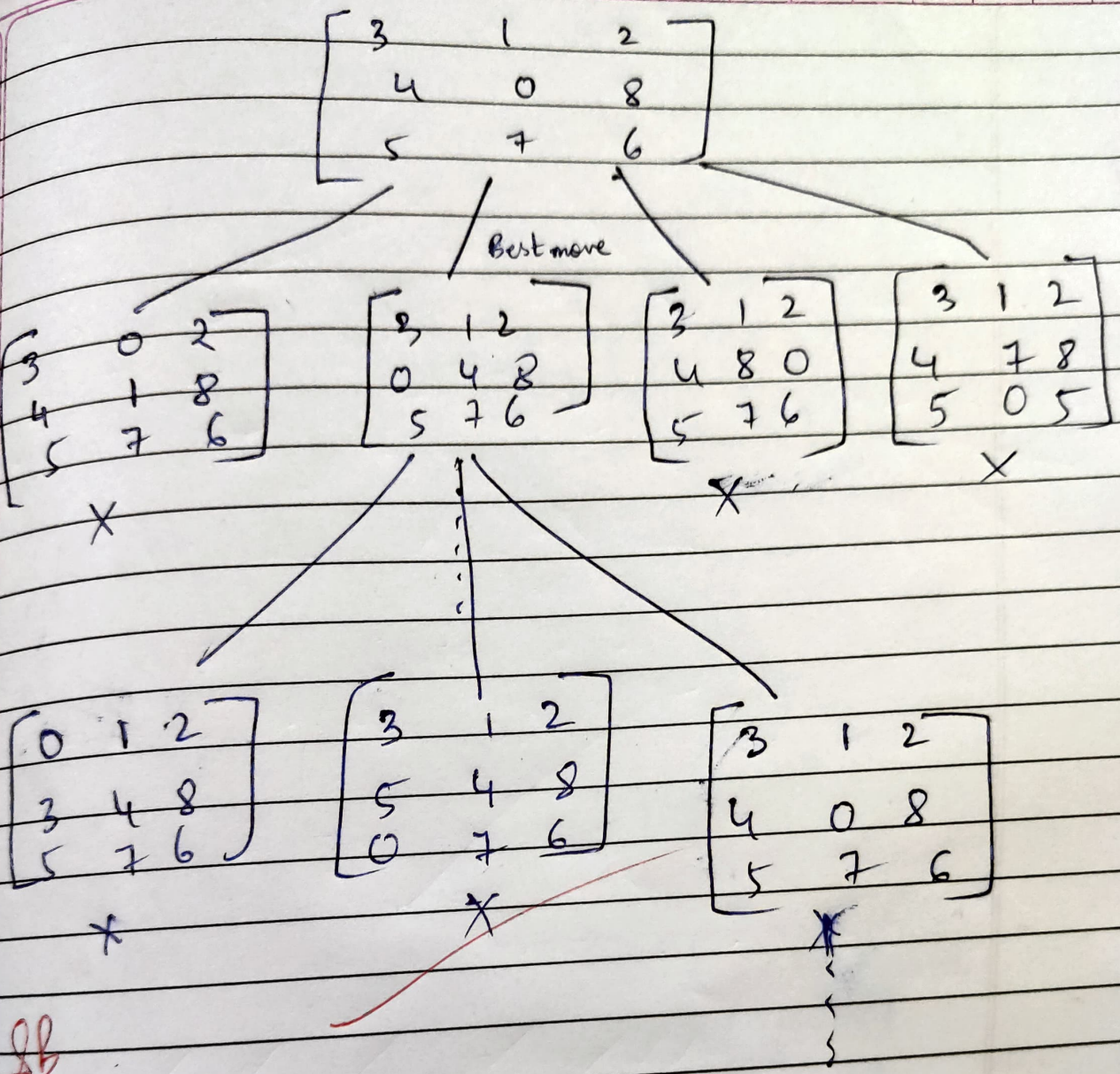
Manhattan Distance (Algorithm)

DATE: PAGE:

Manhattan Distance:

1. Initialize a list which store goal state.
2. Initial state of shuffled puzzle as input from user and store it in a variable.
3. Calculate Manhattan Distance:
   Md = abs(curr x − goal x) + abs(curr y − goal y)
4. Consider a legal move and consider next levels till the goal state is reached using backtracking
5. Using Manhattan distance and priority queue optimize the level at each state and move further based on minimum Manhattan distance.

DATE: PAGE:



Hash