**AI LAB Assignment – 3**

Name: Akshay Tanaji Jadhav

Class: TY CS A

Batch: 02

Roll No: 07

PRN: 12110110

**Problem Statement:** Implementation of Informed strategies.

**Theory**: The 8-puzzle problem is a classic puzzle in which a 3x3 grid is filled with numbered tiles from 1 to 8, along with one empty space. The objective is to rearrange the tiles from a given initial configuration to a specified goal configuration by sliding tiles into the empty space. Only tiles adjacent to the empty space can be moved, and the goal is to achieve the goal configuration in the fewest possible moves. This problem is commonly used as a benchmark for testing search algorithms and artificial intelligence techniques.

**Approach:**

Best First Search (BFS) is an informed search algorithm that explores a graph by expanding the most promising node first, according to some heuristic evaluation function. When applied to the 8-puzzle problem, BFS uses a heuristic to guide the search towards the goal state efficiently. Here's how BFS is applied to solve the 8-puzzle problem:

**Initial State:** Start with the initial configuration of the puzzle.

**Heuristic Evaluation Function:** Define a heuristic evaluation function to estimate the cost of reaching the goal state from any given state. In the case of the 8-puzzle problem, a common heuristic is the Manhattan distance, which calculates the total distance each tile is away from its goal position.

**Priority Queue:** Maintain a priority queue to store the nodes (states) to be explored. Each node contains the current state of the puzzle, its heuristic value, and the cost to reach that state from the initial state.

**Expand Nodes:** Repeat the following steps until a solution is found or the priority queue is empty:

Pop the node with the lowest heuristic value from the priority queue.

Check if the current state is the goal state. If yes, the solution is found.

Otherwise, generate all possible successor states by moving tiles in the puzzle. Evaluate each successor state using the heuristic function and add them to the priority queue.

**Solution Path:** If a solution is found, backtrack from the goal state to the initial state to reconstruct the solution path. This path represents the sequence of moves required to solve the 8-puzzle problem optimally.

**Algorithm:**

function BestFirstSearch(initialState):

Initialize an empty priority queue

Initialize an empty set for visited states

Add the initial state to the priority queue with priority based on heuristic

While the priority queue is not empty:

CurrentState = Remove state with the lowest priority from the priority queue

If CurrentState is the goal state:

Return CurrentState

Add CurrentState to the visited set

Generate all possible successor states from CurrentState

For each successor state:

If successor state is not in the visited set:

Add the successor state to the priority queue with priority based on heuristic

Return "No solution found" if the priority queue is empty

**Output:**

**Initial State:**

**-------------**

**| 1 | 2 | 3 |**

**| | 4 | 6 |**

**| 7 | 5 | 8 |**

**-------------**

**Solution found:**

**-------------**

**| 1 | 2 | 3 |**

**| | 4 | 6 |**

**| 7 | 5 | 8 |**

**-------------**

**-------------**

**| 1 | 2 | 3 |**

**| 4 | | 6 |**

**| 7 | 5 | 8 |**

**-------------**

**-------------**

**| 1 | 2 | 3 |**

**| 4 | 5 | 6 |**

**| 7 | | 8 |**

**-------------**

**-------------**

**| 1 | 2 | 3 |**

**| 4 | 5 | 6 |**

**| 7 | 8 | |**

**-------------**

**Conclusion:**

By using the Best First Search algorithm with an appropriate heuristic function, we can efficiently explore the state space of the 8-puzzle problem and find the optimal solution with fewer computational resources compared to uninformed search algorithms.