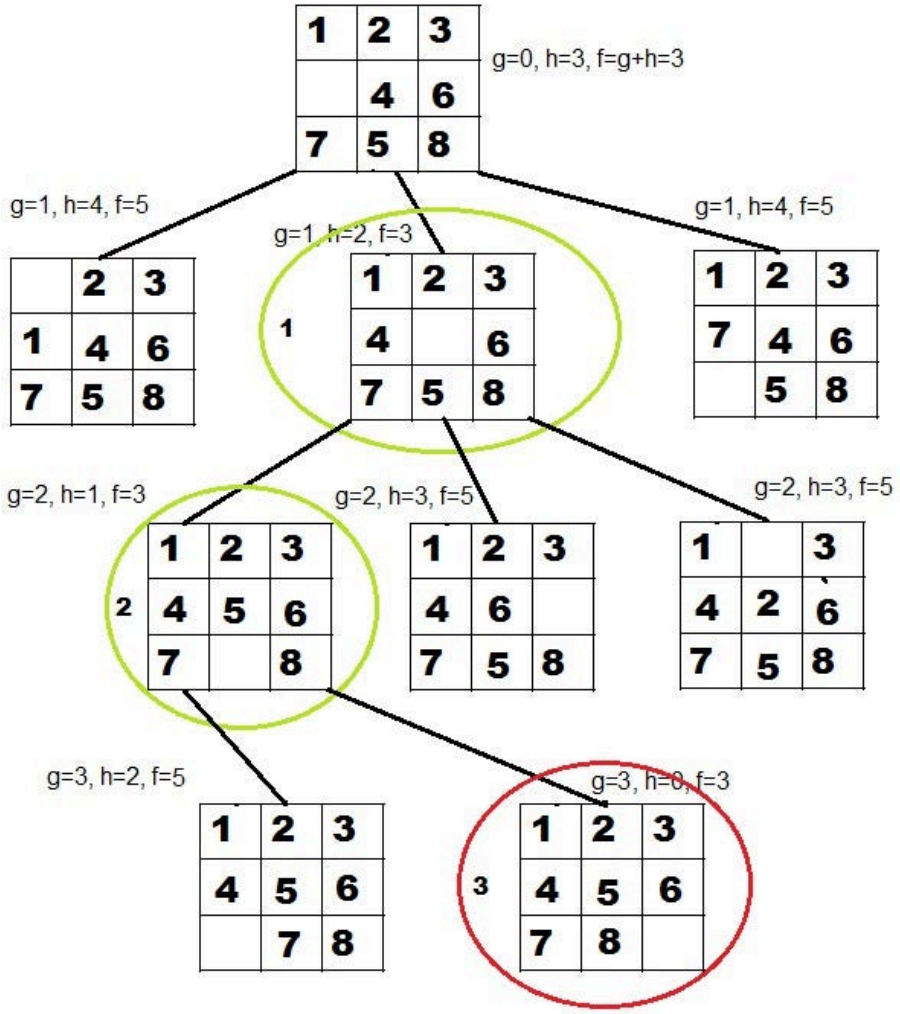
Lab Task 5

A* and Hill Climbing Searches

Task 1: 8 Puzzle Problem

The 8-puzzle problem consists of a 3×3 grid with eight numbered tiles (1-8) and one empty space (0). The tiles can be moved up, down, left, or right into the empty space. The objective is to arrange the tiles in ascending order from left to right, top to bottom, with the empty space in the bottom-right corner.



Write a Python program that implements the A* search algorithm to solve the 8-puzzle problem. Your program should:

1. Take an initial state as input (either hardcoded or user-input).
2. Use A* with a heuristic function (h(n)) to find the optimal solution.
3. Output the sequence of moves (Up, Down, Left, Right) required to reach the goal state.
4. Display the number of moves required and the total number of nodes expanded.

Heuristic Function (h(n))

Use the Manhattan Distance Heuristic: The sum of the distances (rows + columns) of each tile from its correct position.

```
Initial State:
1 2 3
4 0 6
7 5 8
```

```
Move: Right
Move: Down
Move: Left
Move: Up
Goal reached in 4 moves!
Total nodes expanded: 12
```

Note: The program should avoid exploring duplicate states to optimize performance.

Task 2: N-Queen Problems using the Hill climbing search

The N-Queens Problem requires placing N queens on an N × N chessboard such that no two queens attack each other. Queens can attack in the same row, column, or diagonals.

Your task is to implement the Hill Climbing algorithm to find a solution for a given N. The algorithm should use a heuristic function to guide the search towards a solution efficiently.

Given:

- An integer N (size of the chessboard and number of queens).
- A random initial configuration of N queens, where each queen is placed in a random column of its respective row.

Task:

1. Use the Hill Climbing algorithm to iteratively improve the board state.
2. Define a heuristic function (h(n)) that measures the number of conflicts between queens (lower is better).
3. Generate neighboring states by moving a queen to another row in the same column.
4. Select the best neighbor (one with the lowest h(n)) and continue until no better state is found.
5. Print the final board configuration if a solution is found.
6. If the algorithm gets stuck in a local maximum, implement random restarts to increase the chances of finding a solution.

Heuristic Function (`h(n)`)

`h(n)` = Number of **attacking queen pairs** (lower is better).

Example Input:

```
N = 8
```

Example Output (Solution Board for N=8):

```
_ Q _ _ _ _ _ _
_ _ _ _ Q _ _ _
_ _ _ _ _ _ Q _
Q _ _ _ _ _ _ _
_ _ _ _ _ Q _ _
_ _ Q _ _ _ _ _
_ _ _ _ _ _ _ Q
_ _ _ Q _ _ _ _
Solution found!
```