AI Project

N-Puzzel

	Name	Id	Dept & LvL	Grade
1	رحاب نادر جلال علي	20210324	Is _ lvl4	
2	ريم سيد محمود	20210347	It _ lvl4	
3	ريم كمال الدين عبد العزيز	20210349	Is _ lvl4	
4	مي هاني محمد جمال	20210978	Is _ lvl4	
5	ميار احمد فتحي محمد	20210979	Is _ lvl4	
6	يمنى محمد عزت امام	20211055	Is _ lvl4	

Project Link: https://github.com/RehabNader003/N-Puzzle_Ai_Project.git

Introduction and overview:

❖ **Project overview:**

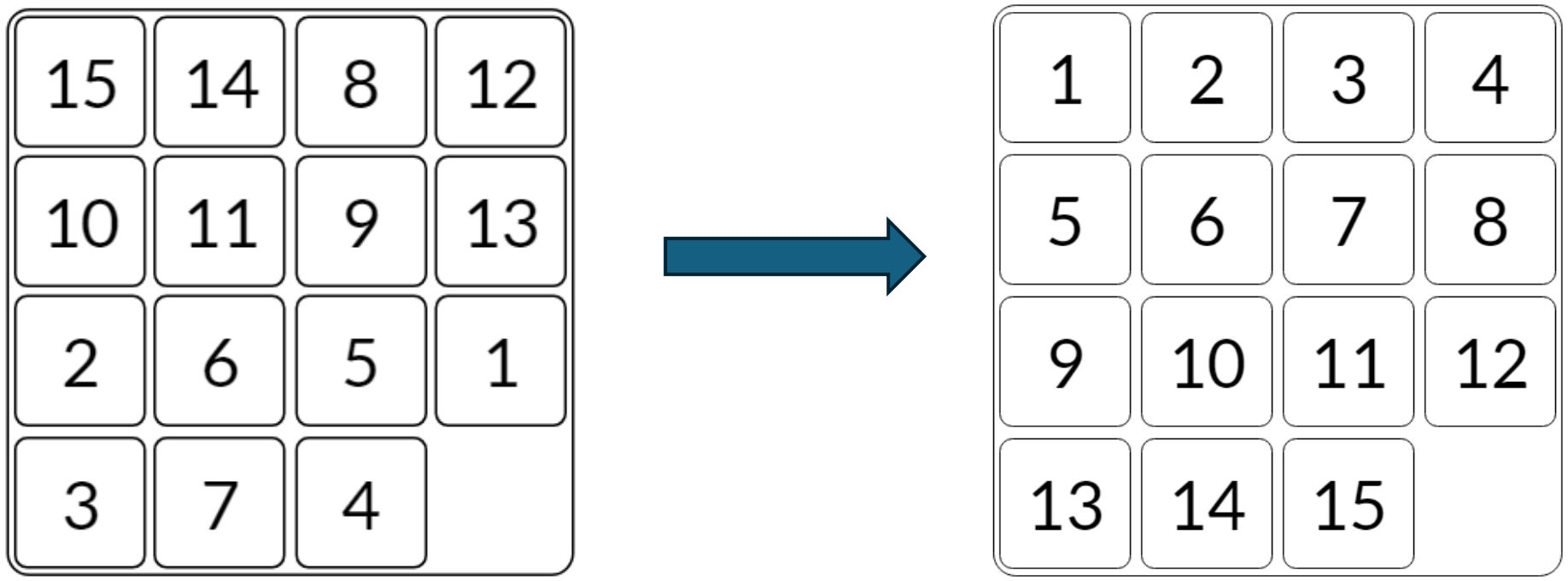
- The N-Puzzle Solver is a graphical application designed to tackle the intriguing challenge of solving N-Puzzles, a class of sliding puzzle problems. The application provides an interactive and user-friendly environment, allowing enthusiasts to explore and solve puzzles of various sizes, such as the classic 8-Puzzle, 15-Puzzle, and even larger configurations like the 24-Puzzle. The N-Puzzle Solver project offers a dynamic and educational platform for solving N-Puzzles. With its interactive features, animated solving process, and support for various heuristic functions, the project provides a captivating experience for users interested in puzzles and algorithmic problem-solving.

- **Similar app**: Next-puzzle (https://igorgarbuz.github.io/n-puzzle/)

What is the N-Puzzel?

- The puzzle exists in various sizes, such as the smaller 3x3 "8 Puzzle" and the larger 5x5 "24 Puzzle". In general, these sliding puzzles are collectively called the "n-puzzle" (also called the "(n^2-1)-puzzle")placed on a grid square , with one tile empty. The goal of the puzzle is to slide the tiles around to reach the solved state

As Ex. 15 Puzzle



Initial state

Goal state

Academic publications:

1-Evaluating Search Algorithms for Solving n-Puzzle

https://sumitg.com/assets/n-puzzle.pdf

- Author: Sumit Gupta
- Link: Evaluating Search Algorithms for Solving n-Puzzle
- Summary: The document likely discusses the evaluation of various search algorithms in the context of solving the n-puzzle problem. It may include empirical comparisons and analyses of different algorithms.
- Heuristic Search Viewed as Path Finding in a Graph

2- Heuristic search viewed as path finding in a graph

https://www.sciencedirect.com/science/article/abs/pii/000437027090007X

- Source: ScienceDirect
- Link: Heuristic search viewed as path finding in a graph
- Summary: This source appears to approach heuristic search as a path-finding problem in a graph.
- It might discuss the theoretical foundations and applications of heuristic search algorithms.
- Pedagogical Possibilities for the N-Puzzle Problem

3- Pedagogical Possibilities for the N-Puzzle Problem

Pedagogical Possibilities for the N-Puzzle Problem

- Source: Gettysburg College
- Link: Pedagogical Possibilities for the N-Puzzle Problem
- Summary: The content likely explores educational aspects of the n-puzzle problem, discussing
- how it can be used for pedagogical purposes, possibly in computer science education.
- An Alternative Solution to N-Puzzle Problem

4- An alternative solution to n-puzzle problem

https://www.researchgate.net/publication/327595535_An_alternative_solution_to_n-puzzle_problem

- Source: ResearchGate
- Link: An alternative solution to n-puzzle problem
- Summary: This source might present an alternative solution or approach to solving the n-puzzle
- problem, potentially introducing novel ideas or algorithms.
- Parallel N-Puzzle Solver in Haskell

5- Parallel N-Puzzle Solver in Haskell

https://www.cs.columbia.edu/~sedwards/classes/2021/4995

- **fall/reports/N-Puzzle.pdf**
- Author: Unknown (based on the link)
- Link: Parallel N-Puzzle Solver in Haskell
- Summary: This source could discuss the implementation of a parallel solver for the n-puzzle
- problem using Haskell, providing insights into parallel computing aspects.

Features

1- User-Friendly Interface:

- The PuzzleSolverGUI offers an intuitive interface for users to set up the puzzle board, choose solving options, and interact with the solving process.

2- Breadth-First Search (BFS) Algorithm:

- The application utilizes the Breadth-First Search algorithm, a fundamental graph traversal algorithm, to systematically explore possible states and find optimal solutions to the puzzles.

3- Heuristic Functions:

- Users can choose from various heuristic functions, including Manhattan distance, Hamming distance, Euclidean distance, and Linear Conflict, to guide the solving process and explore different solving strategies.

4- Interactive Solving Process:

- The application provides an animated solving process, allowing users to visualize the steps taken by the algorithm in real-time. Colorful and dynamic puzzle board animations enhance the solving experience.

5- Solution Information:

- Users can access detailed information about the solved puzzle, including the number of steps taken, elapsed time, and additional statistics.

6- State Exploration:

- Explore different solving states and compare the efficiency of heuristics through the "get_states" feature, providing insights into the performance of various solving strategies.

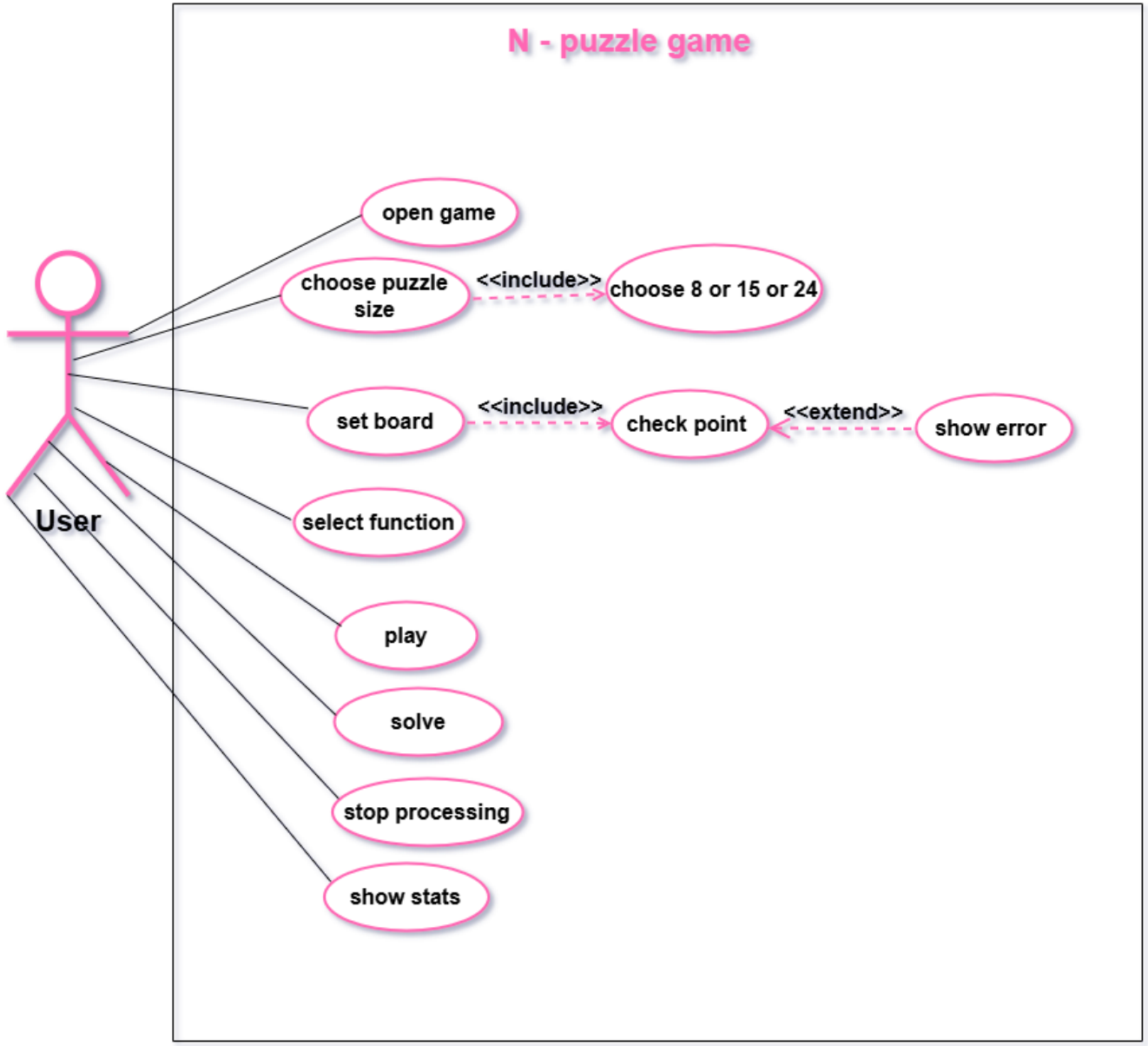
7- Dynamic Controls:

- The application dynamically enables or disables controls based on the current state of the solving process, ensuring a smooth and controlled user experience.

8- Stop and Resume:

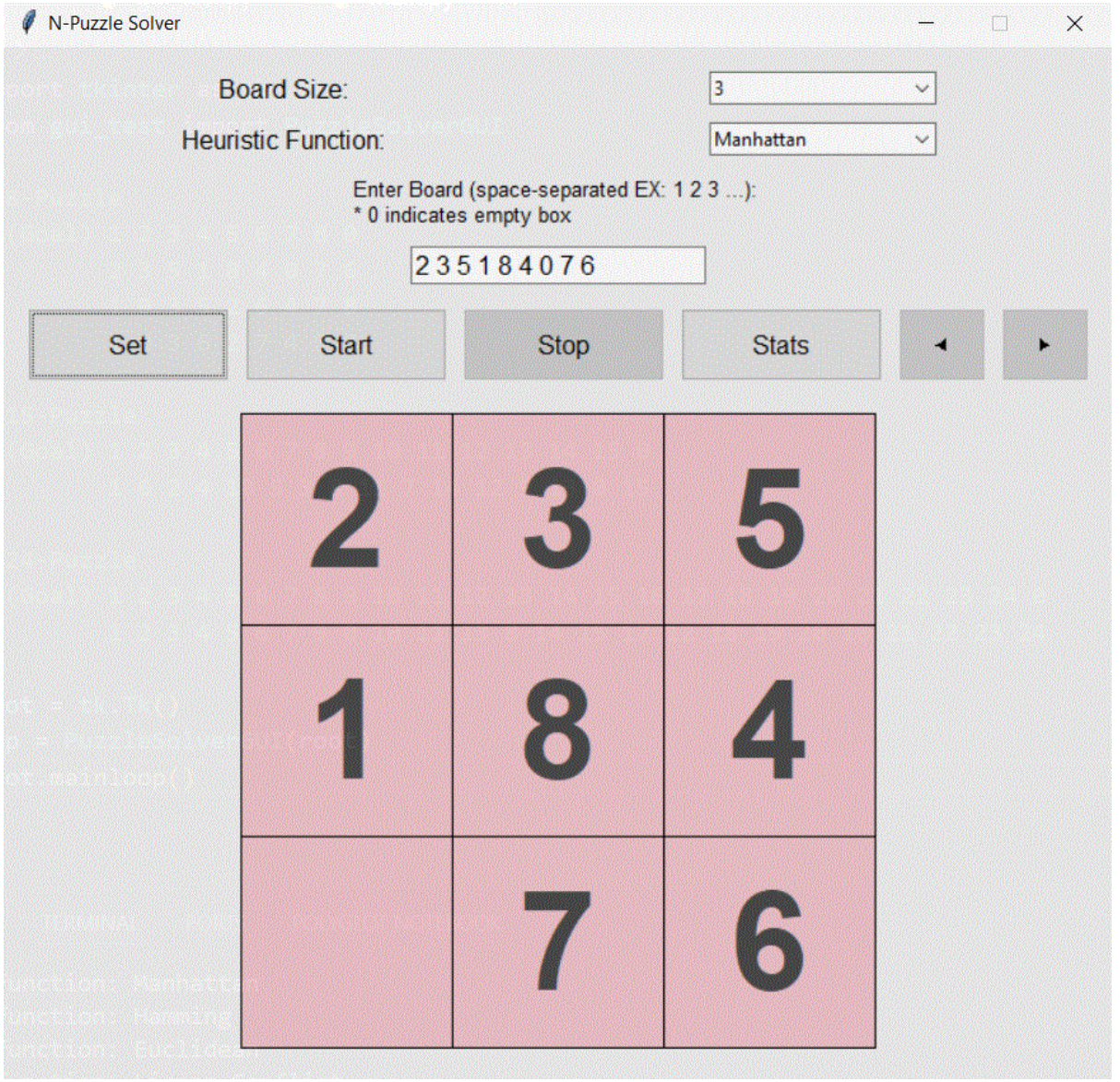
- Permits users to stop the solving process and resume or explore the solution at their own pace

Use case diagram:

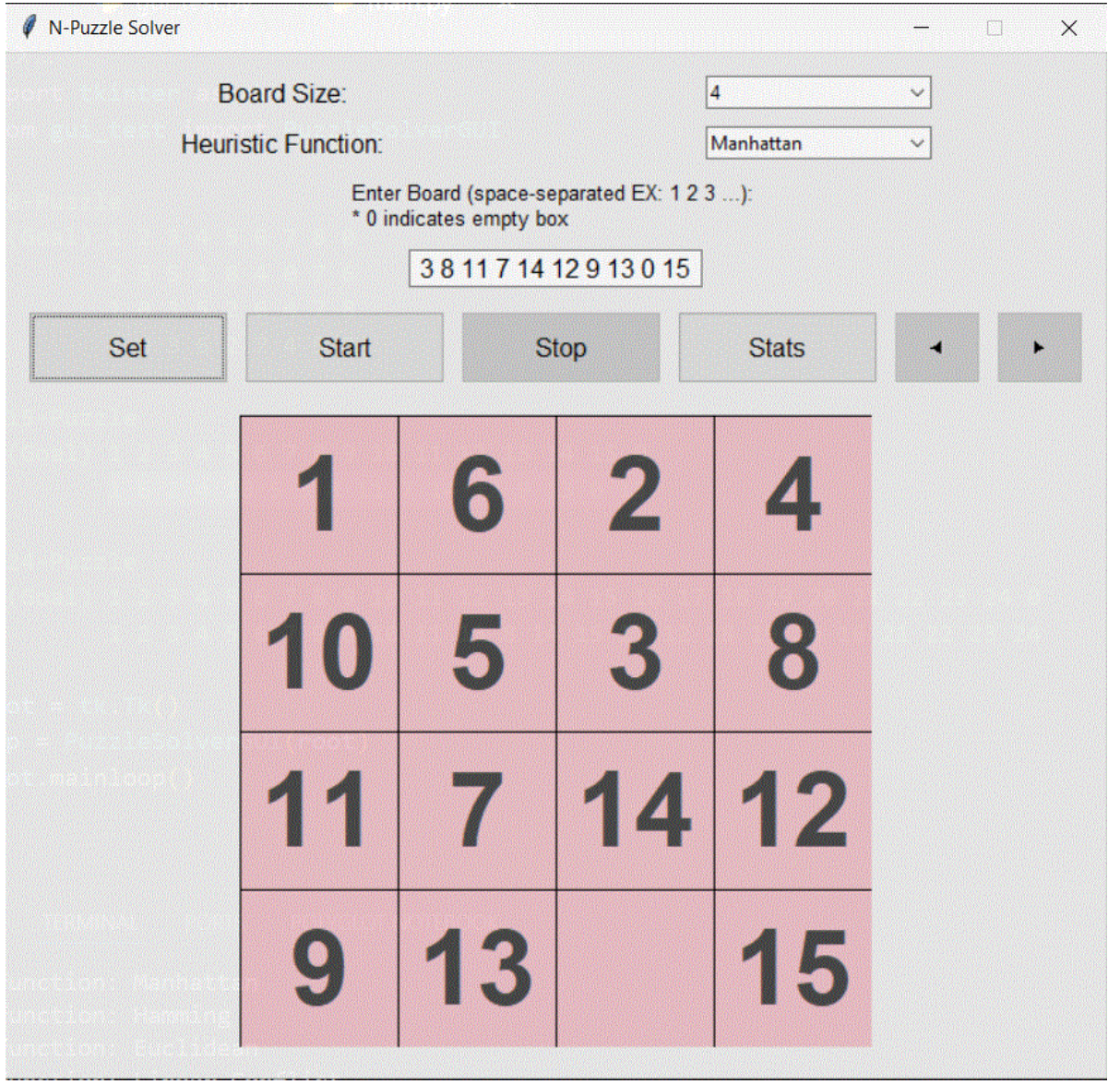


App GUI:

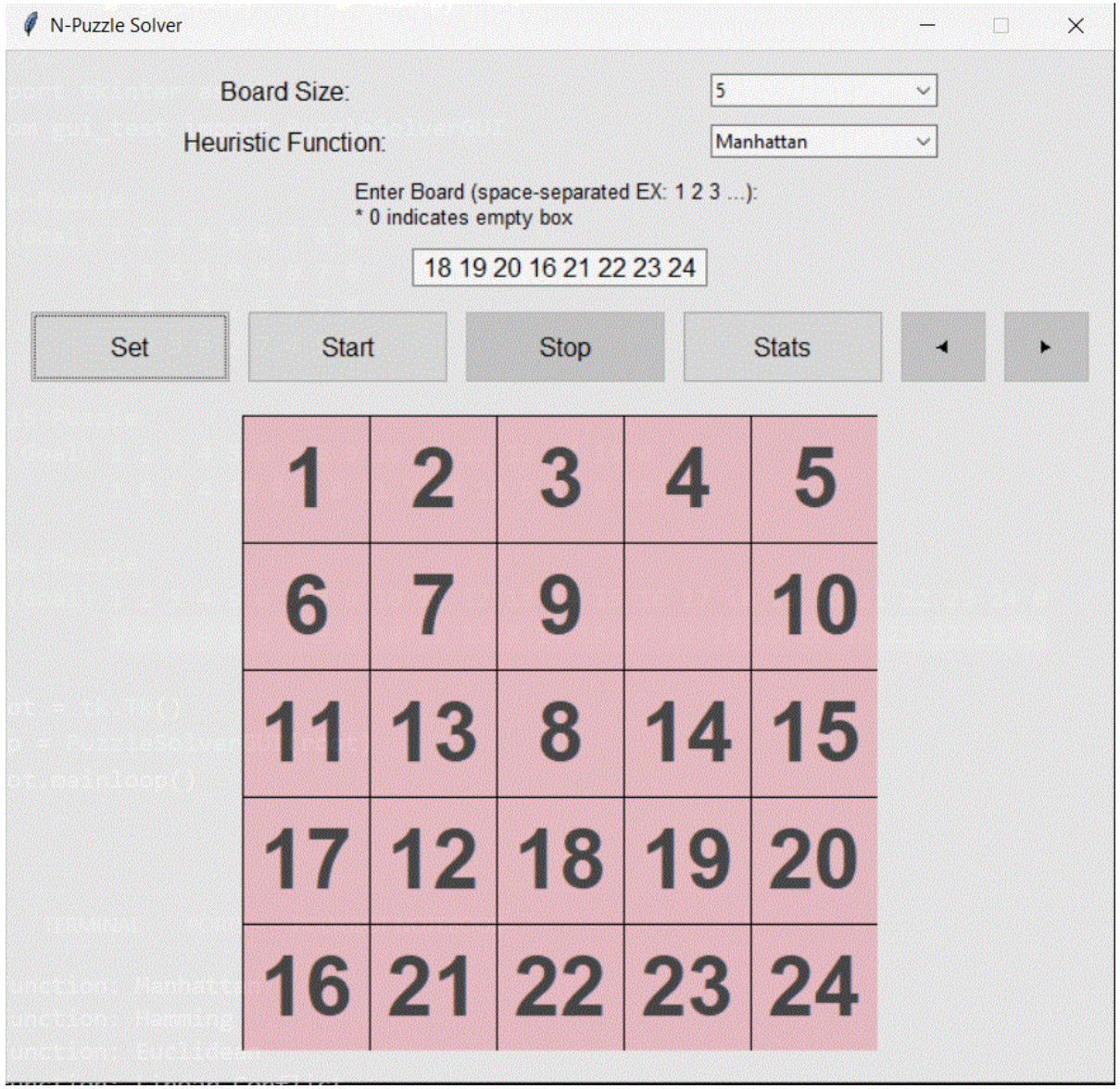
- **3*3 board**


N-Puzzle Solver
Board Size:
3
Heuristic Function:
Manhattan
Enter Board (space-separated EX: 1 2 3 ...):
* 0 indicates empty box
2 3 5 1 8 4 0 7 6
Set
Start
Stop
Stats
2 3 5
1 8 4
7 6

- **4*4 board**


N-Puzzle Solver
Board Size:
4
Heuristic Function:
Manhattan
Enter Board (space-separated EX: 1 2 3 ...):
* 0 indicates empty box
3 8 11 7 14 12 9 13 0 15
Set
Start
Stop
Stats
1 6 2 4
10 5 3 8
11 7 14 12
9 13 15

- **5*5 board**


N-Puzzle Solver
Board Size:
5
Heuristic Function:
Manhattan
Enter Board (space-separated EX: 1 2 3 ...):
* 0 indicates empty box
18 19 20 16 21 22 23 24
Set
Start
Stop
Stats
1 2 3 4 5
6 7 9 10
11 13 8 14 15
17 12 18 19 20
16 21 22 23 24

Guide:

1- User should select board size and select which heuristic function to use.

2- User should provide valid n-puzzle as an input.

3- User should press set button to update the game board with provided input.

4- User can press the start button to start processing or he can press stats button

which show number of visited node and number of moves starting from the initial node and ending with goal node.

5- After finishing processing, the user can navigate through solution path.

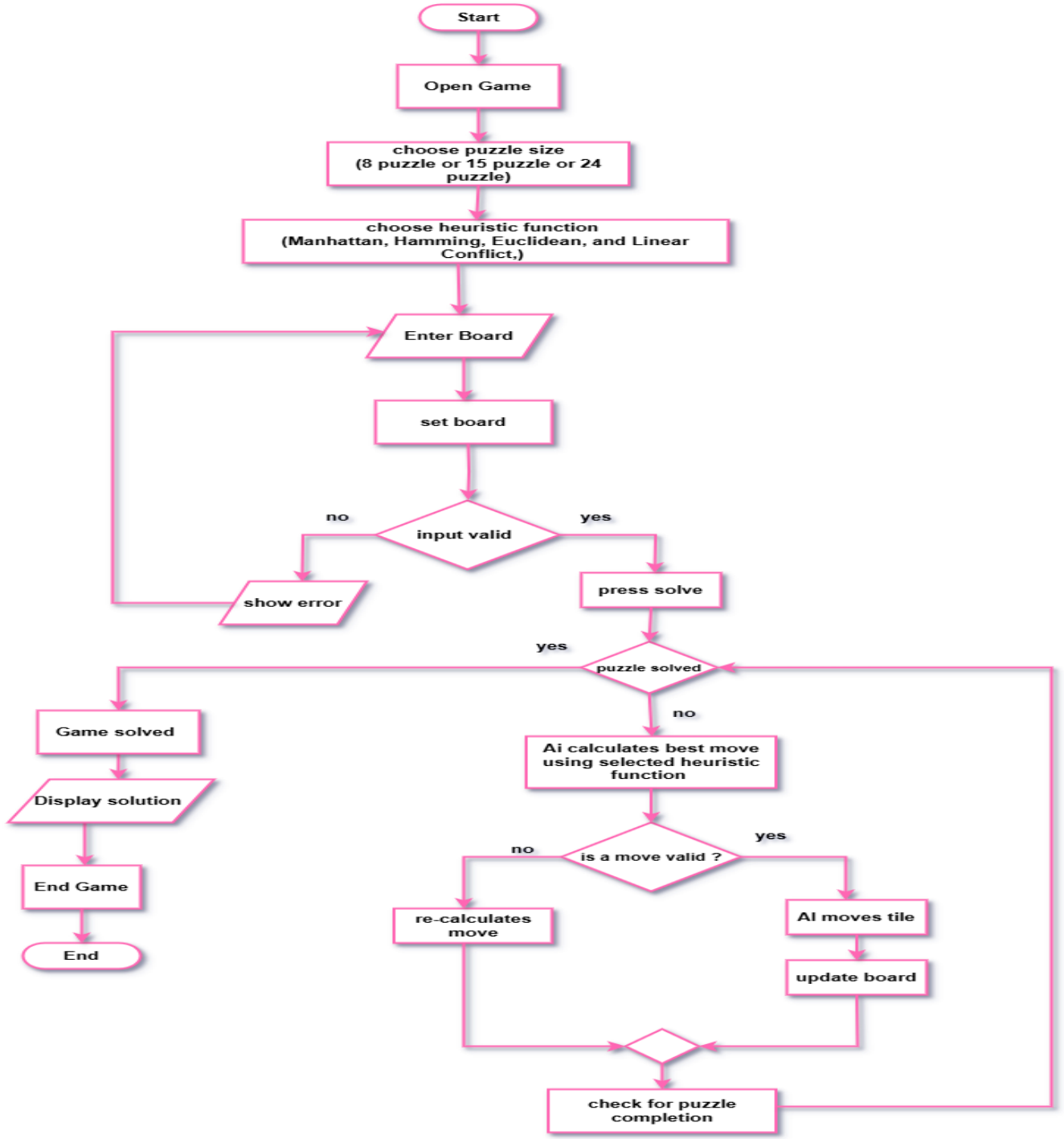
6- User can stop processing immediately after pressing the stop button.

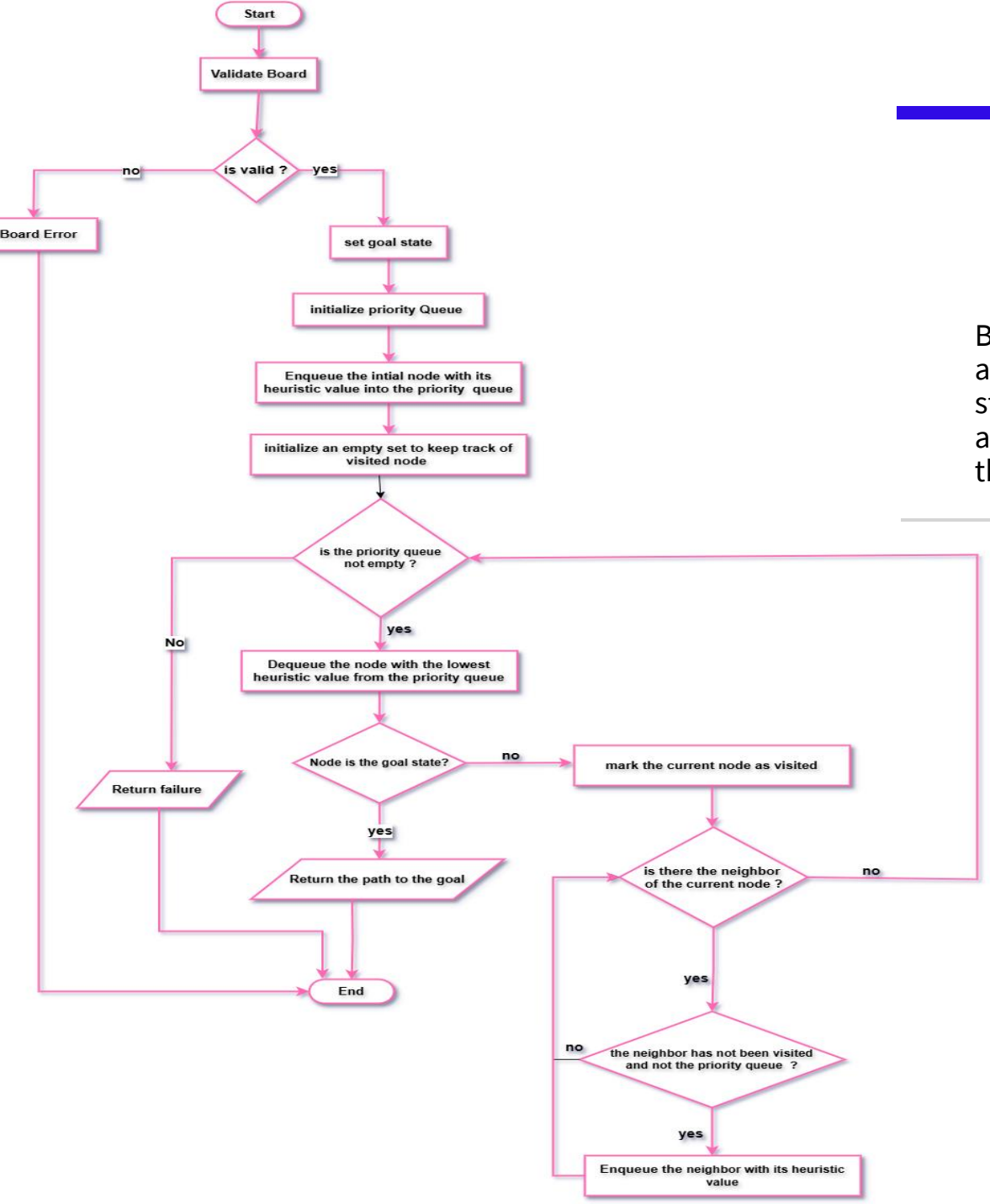
Constraints:

1. Number of elements should be equals to n*n (n = board)
2. Each number separated with whitespace
3. 0 indicate empty tile
4. input should not include strings

Flowcharts Diagram

- **Setup**: Input the puzzle size, board, and heuristic.
- **Solve**: The AI calculates moves using a heuristic function.
- **Validate Moves**: Ensure each move is correct and update the board.
- **Completion**: Check if the puzzle is solved and display the solution.This ensures the game progresses step-by-step toward solving the puzzle efficiently.





— Applied Algorithms

- Best-First Search algorithm

Best-First Search is a versatile algorithm employed in computer science and artificial intelligence for graph or tree traversal. It efficiently explores these structures by selecting nodes based on a heuristic evaluation function. The algorithm's primary goal is to find solutions or reach a goal state while minimizing the search space.

Key Concepts:

Heuristic Function:

Relies on a heuristic function to assess the desirability of expanding nodes. This function estimates the cost or distance from the current node to the goal, guiding the search toward promising paths.

Priority Queue:

Nodes are expanded based on their heuristic values.
Utilizes a priority queue to manage nodes, with lower heuristic values prioritized for exploration.

Exploration Strategy:

An informed search algorithm leveraging domain-specific knowledge.
Focuses on exploring paths deemed most promising, in contrast to uninformed algorithms.

Completeness and Optimality:

Depends on the admissibility and consistency of the heuristic function.
If the heuristic is admissible and consistent, Best-First Search is both complete (finds a solution if one exists) and optimal (finds the least-cost solution).

Applications:

Widely applied in pathfinding (e.g., robotics), puzzle solving, game playing (chess), and network routing.

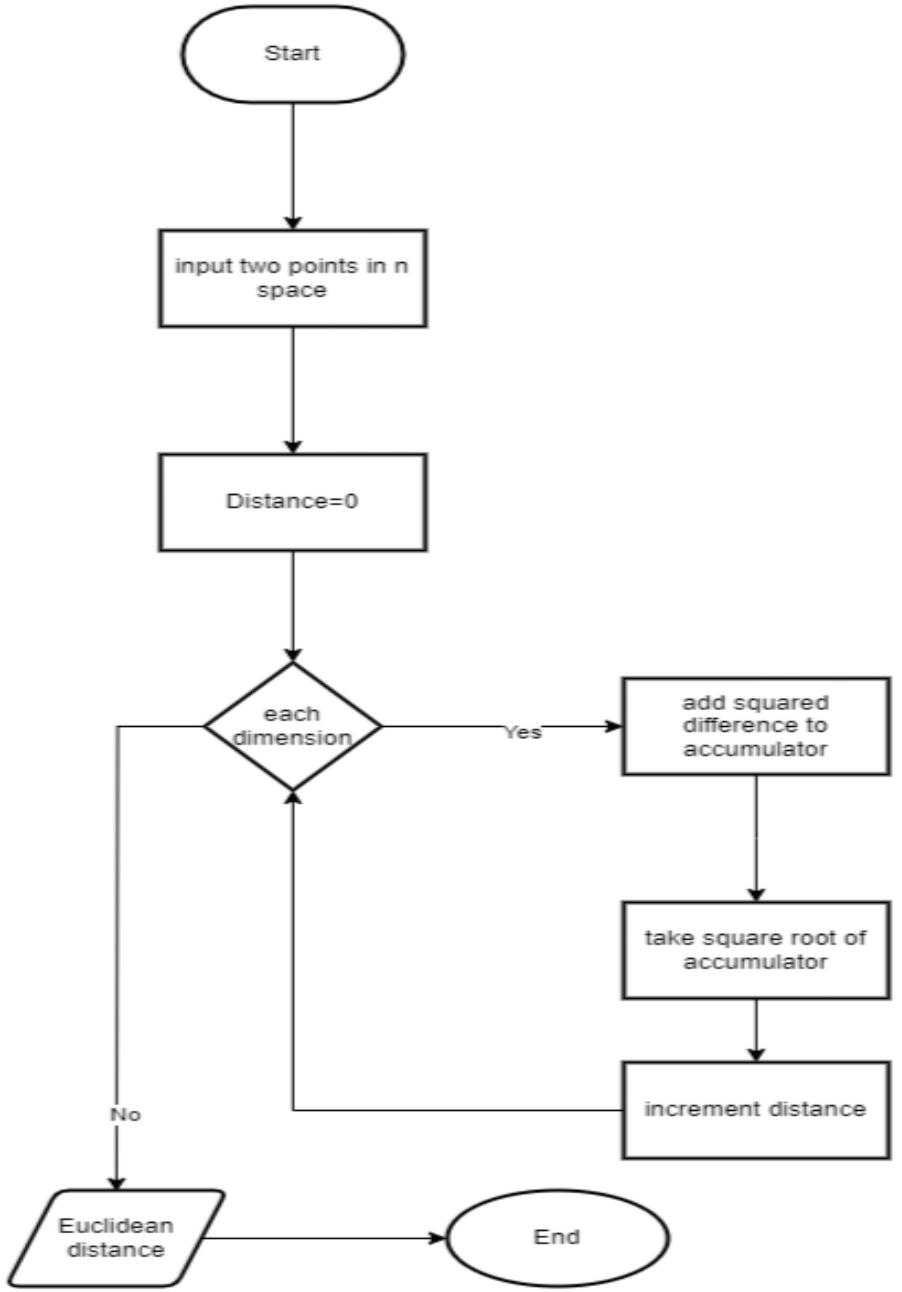
Flow chart(Euclidean Function):

Definition: Euclidean distance is a measure of the straight-line distance between two points in Euclidean space.

Formula: For two points (x1, y1) and (x2, y2), the Euclidean distance is calculated as:

Distance= $\sqrt{(x2-x1)2+(y2-y1)2}$

Application: Euclidean distance is often used in geometric problems, image processing, and clustering algorithms.

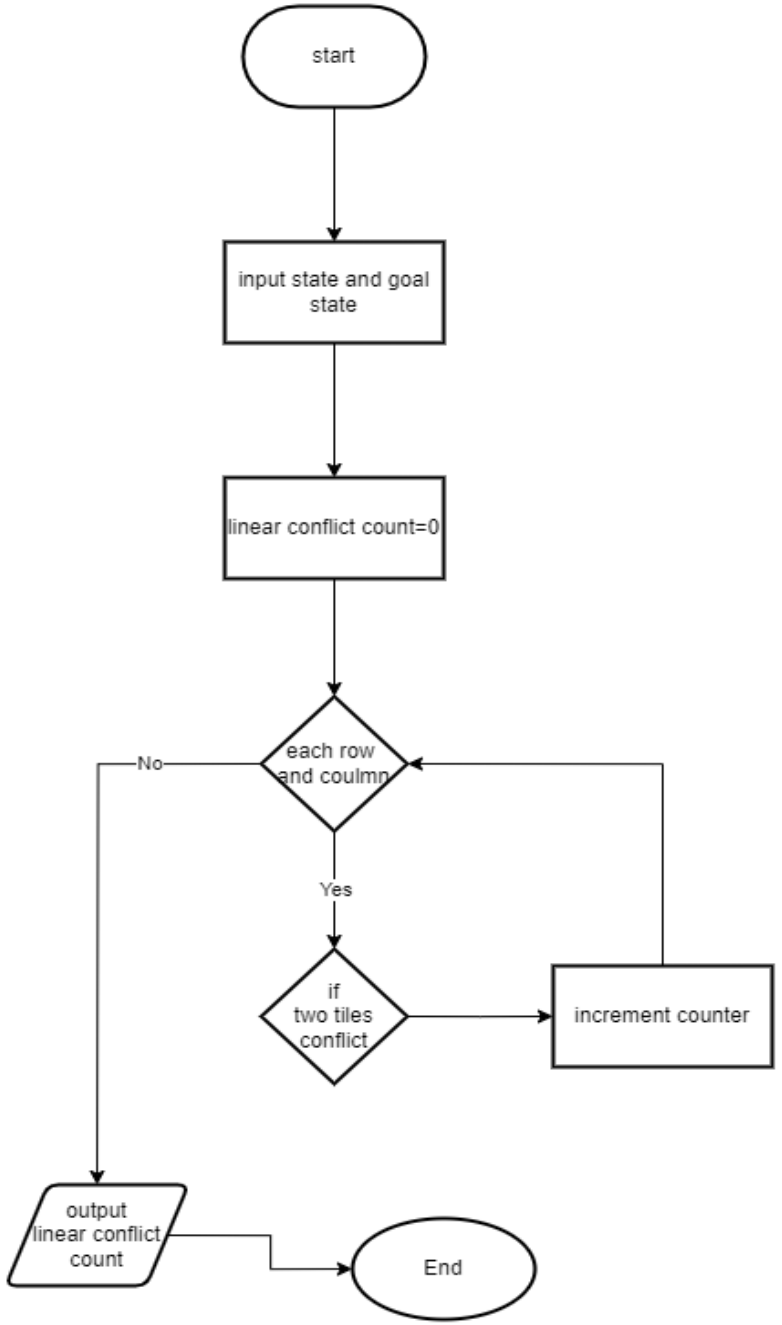


Flow chart(linear conflict function):

- **Definition**: Linear conflict is a concept used in the context of the N-puzzle problem (sliding puzzle games) to enhance heuristic estimation. It takes into account conflicts that arise when two tiles that should be in a particular order are in the opposite order.

- **Explanation**: In the context of the A* search algorithm for the N-puzzle problem, linear conflict extends Manhattan distance by considering conflicts in both row and column directions. If two tiles are in their goal positions but are in the same row or column and their goal positions are in the opposite order, a linear conflict is present.

- **Application**: Linear conflict is used to improve the efficiency of heuristic functions in the A* algorithm for solving sliding puzzle problems.

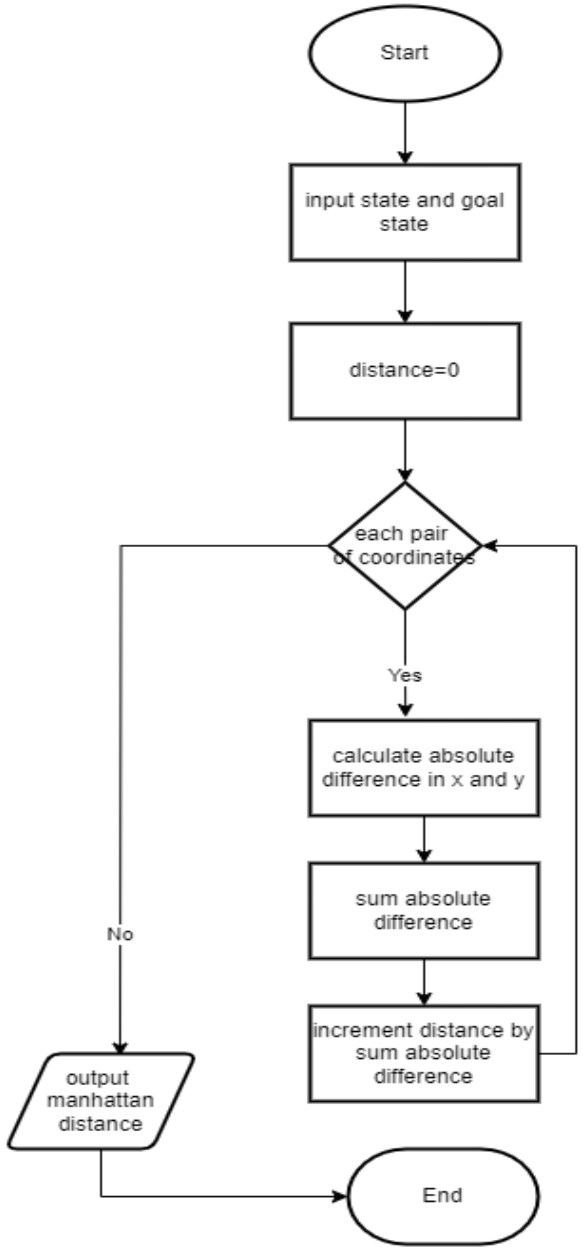


Flow chart(Manhattan distance function):

- **Definition**: Manhattan distance, also known as L1 distance or taxicab distance, is the sum of the absolute differences between the coordinates of two points.

- **Formula**: For two points (x1, y1) and (x2, y2), the Manhattan distance is calculated as:
- **Distance**=|x2−x1|+|y2−y1|

- **Application**: Manhattan distance is commonly used in pathfinding algorithms and grid-based problems where movement can only occur along grid lines.

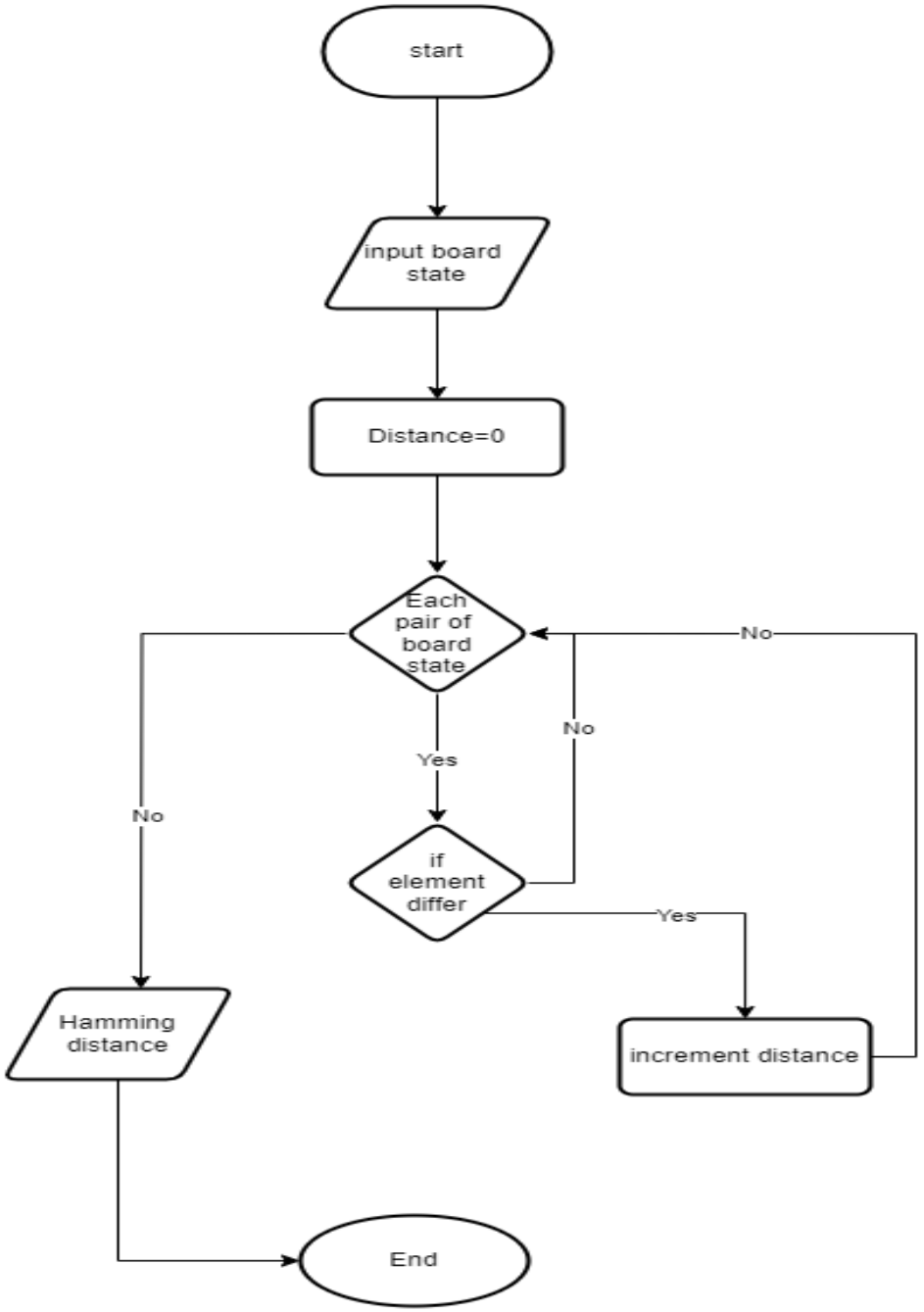


Flow chart(hamming distance function):

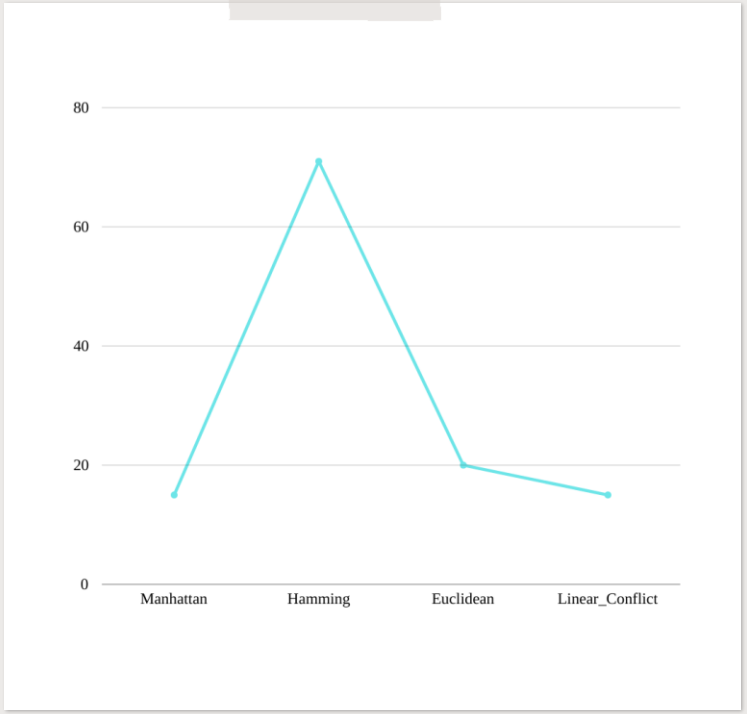
- **Definition**: Hamming distance measures the number of positions at which corresponding bits are different between two equal-length strings.

- **Formula**: For two strings of equal length, the Hamming distance is the count of positions where the characters differ.

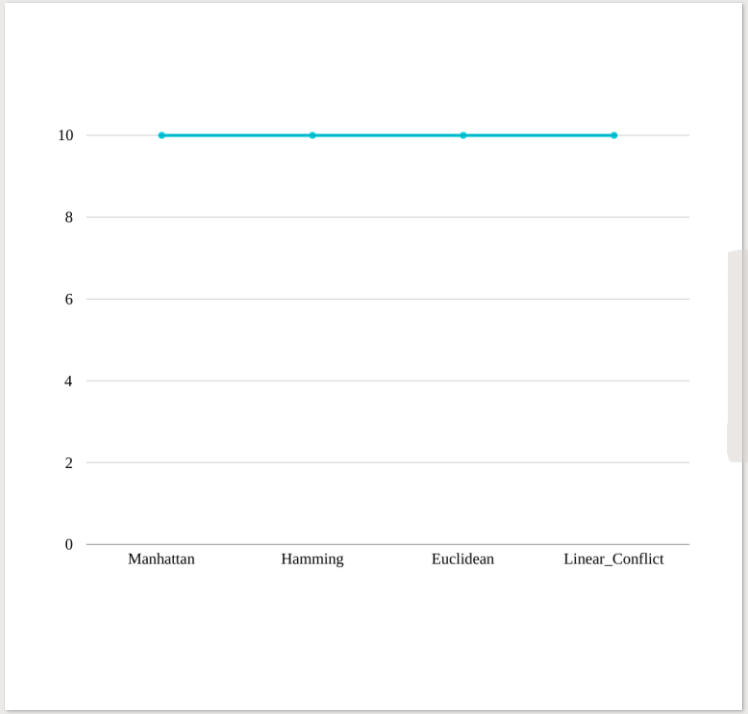
- **Application**: Hamming distance is often used in error detection and correction codes, DNA sequence analysis, and information retrieval



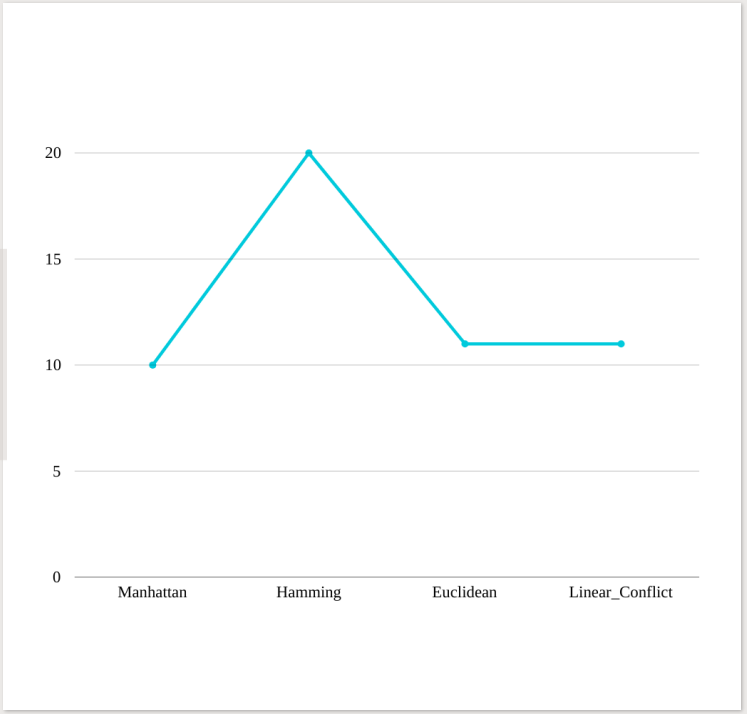
4*4



5*5

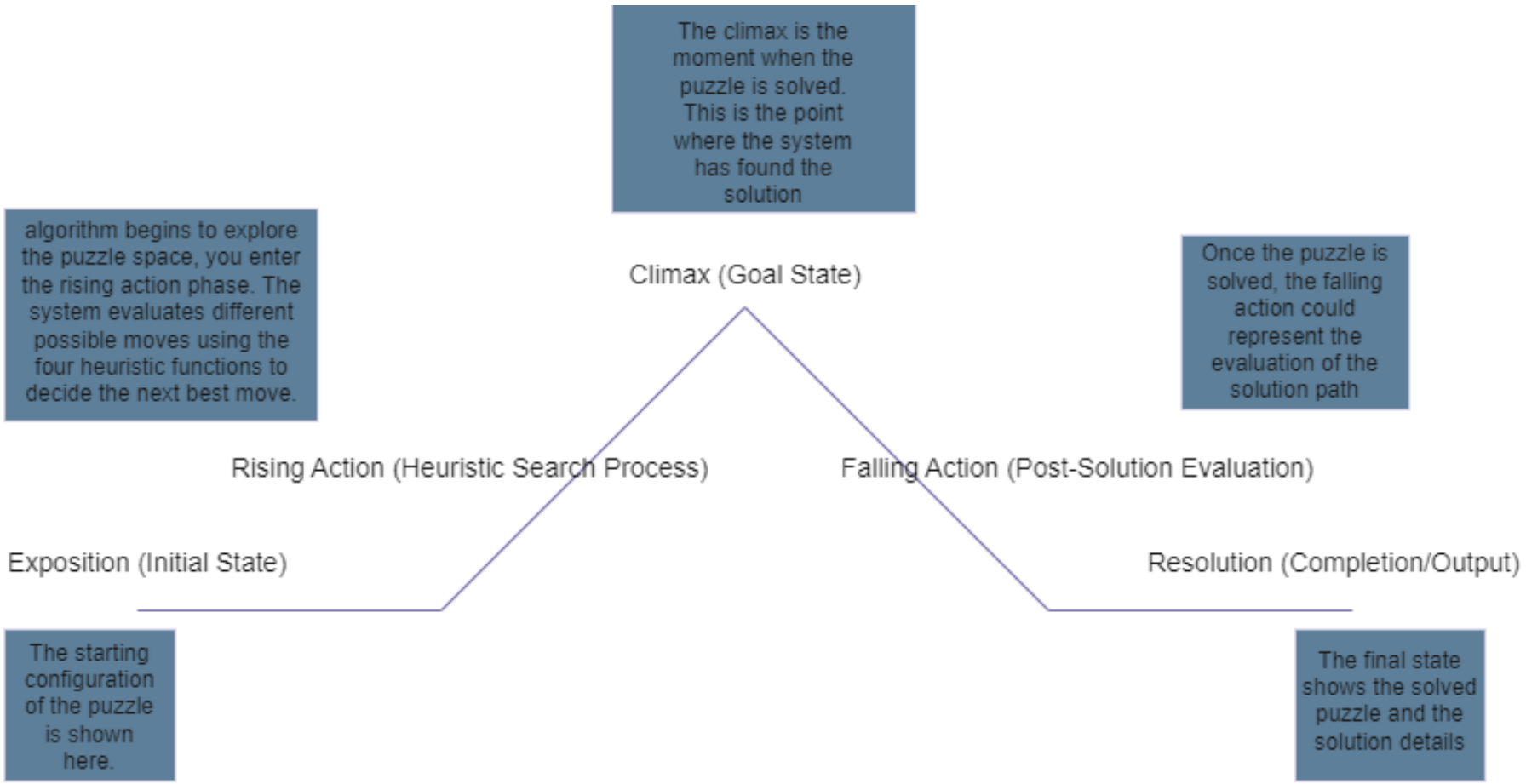


3*3

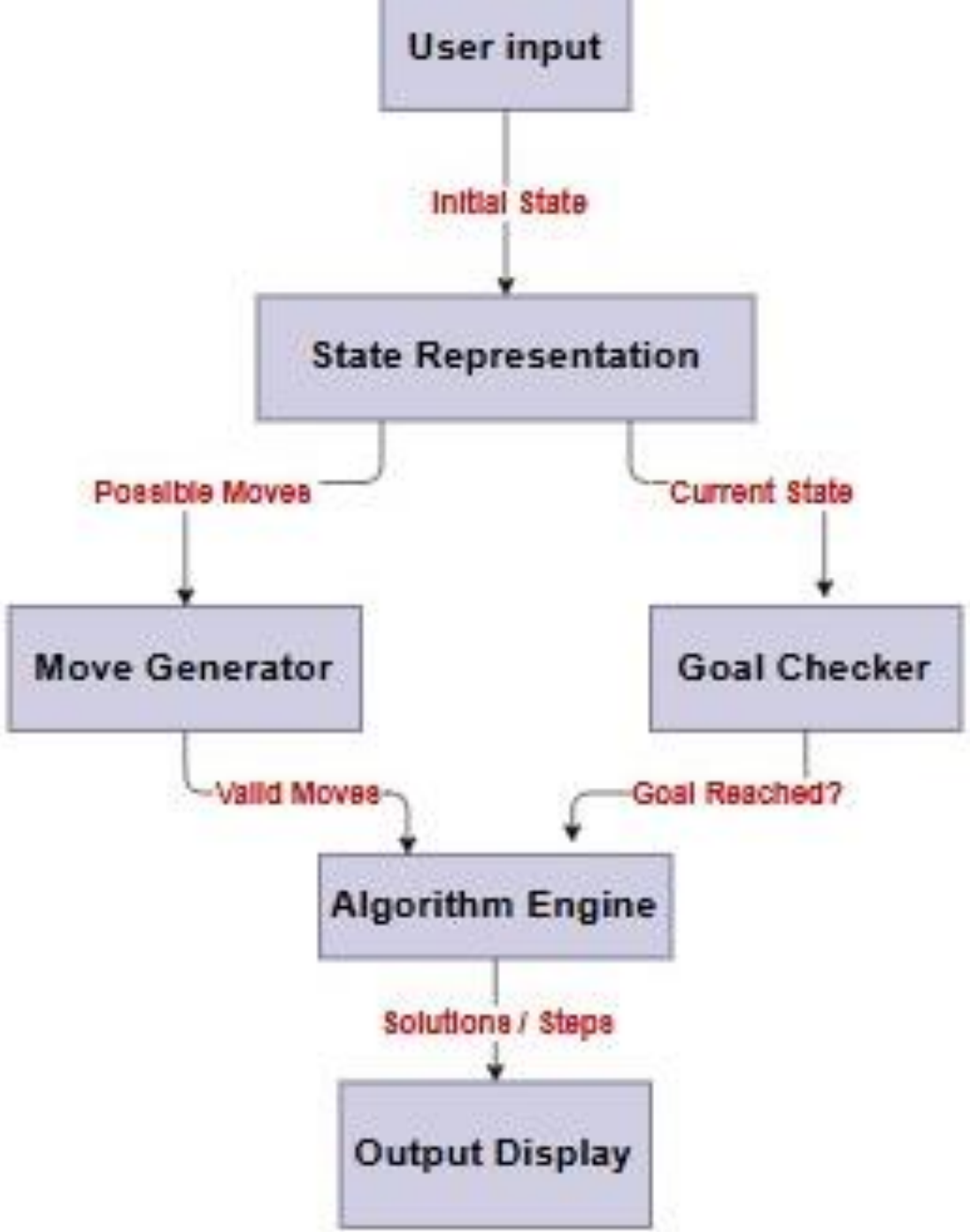


plots show number of visited nodes for every heuristic:

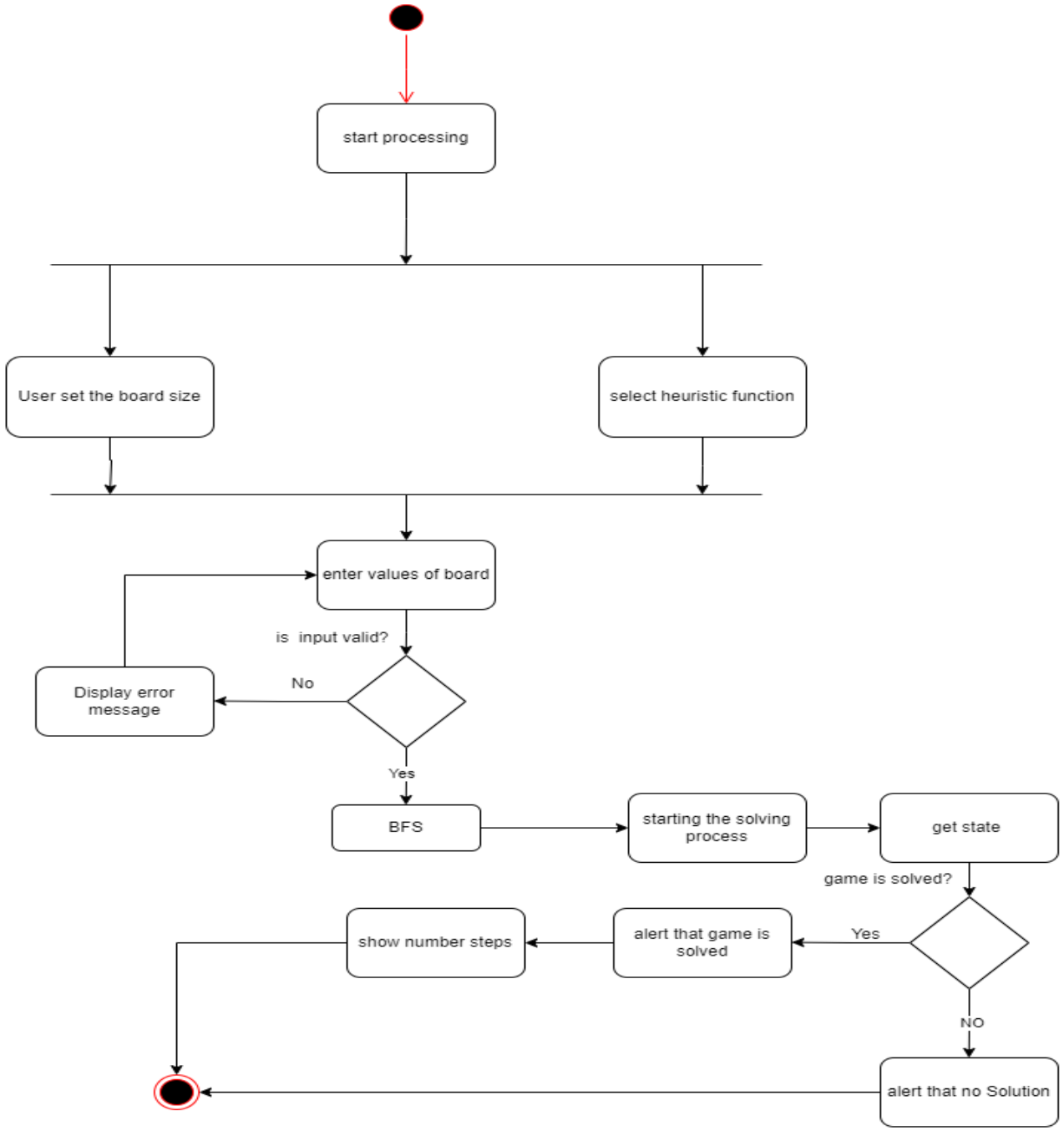
Plot diagram:



Block diagram:



Activity Diagram:



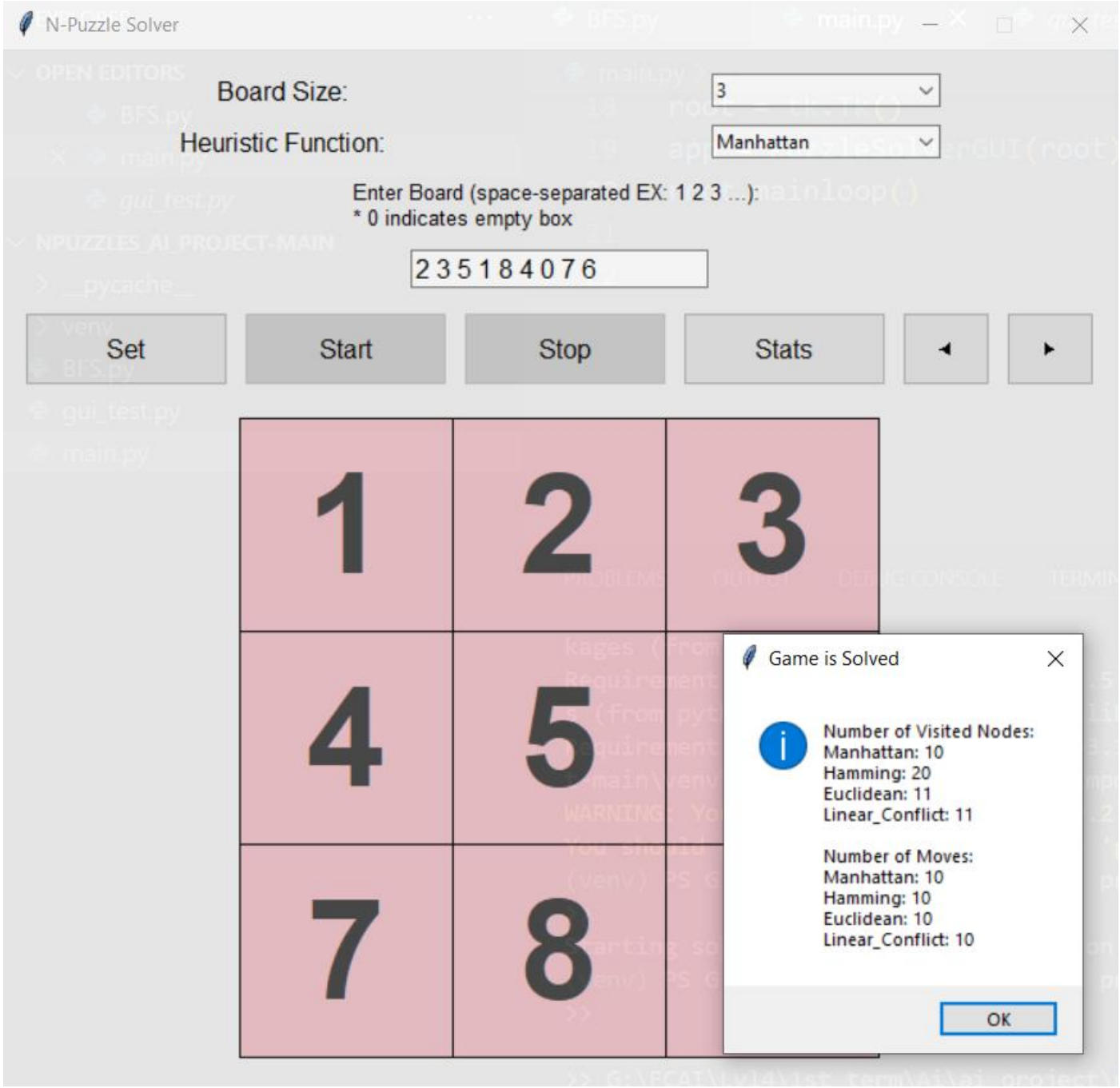
Experiments & Results:

in example we have seen that :

hamming distance has the lowest performance compared to other heuristics

Manhattan distance has the highest performance

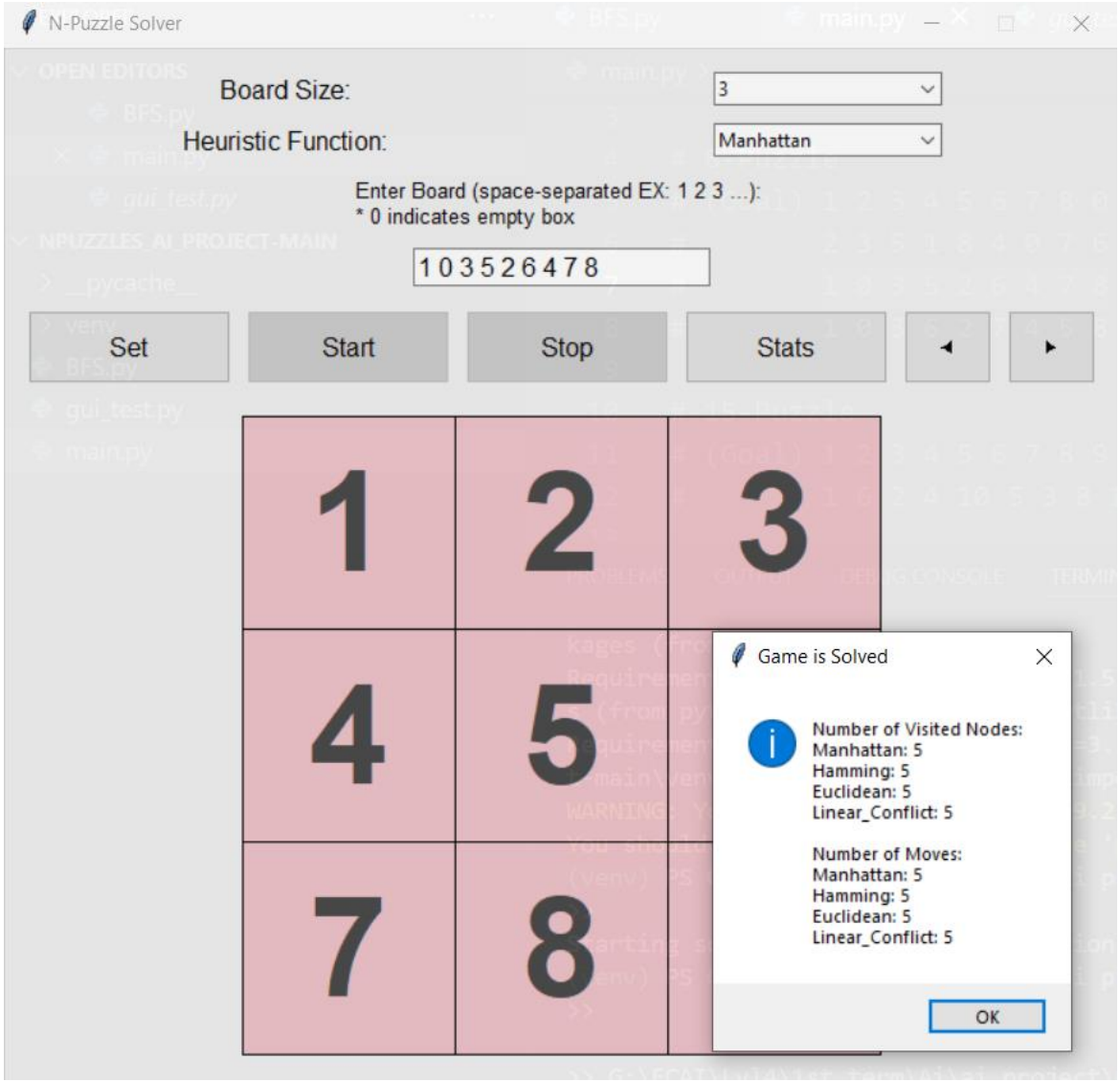
linear conflict and **Euclidean distance** perform equally



in this example we have seen that:

all heuristic functions perform equally

in (1-0-3-5-2-6-4-7-8)



Analysis, Discussion:

Applicability: Best-First Search is applicable to a wide range of problems, including pathfinding, puzzle-solving, game-playing, and optimization problems.

Disadvantages of Best-First Search:

Completeness Depends on Heuristic Quality: The completeness of BFS depends on the quality of the heuristic function. If the heuristic is poorly designed or not admissible, BFS might not find a solution even if one exists.

Memory Usage: Best-First Search can be memory-intensive, especially when dealing with large search spaces. The algorithm may need to store a large number of nodes in memory, which can be a limitation in resource-constrained environments.

Sensitivity to Heuristic Quality: The performance of BFS is highly dependent on the accuracy of the heuristic function. If the heuristic is not well-tuned to the problem, the algorithm may not efficiently explore the most promising paths.

Insights into Algorithm Behavior:

Heuristic Influence: The behavior of BFS is heavily influenced by the heuristic function. A well-designed heuristic can significantly improve the algorithm's efficiency by guiding it toward the most promising solutions.

Exploration vs. Exploitation: BFS strikes a balance between exploration (searching new paths) and exploitation (choosing the best-known paths). The algorithm tends to prioritize paths that appear more promising based on the heuristic information.

Optimality Guarantee: If the heuristic is admissible, BFS provides an optimality guarantee, ensuring that the solution found is the best possible given the available information.

Future Work::

Heuristic Improvement: Experimenting with different heuristic functions or refining the existing one could be beneficial. The goal is to design a heuristic that provides more accurate estimates of the true cost, guiding the search more effectively.

Memory Optimization: Considering modifications to reduce the algorithm's memory footprint, such as implementing techniques like iterative deepening or using more memory-efficient data structures.

Dynamic Heuristics: Implementing a heuristic that adapts or changes dynamically during the search based on the characteristics of the problem could improve the algorithm's performance in certain scenarios.

Parallelization: Exploring possibilities for parallelizing the search process could be considered to make better use of available computing resources.

Resources:

Algorithm

- -> Best First Search:
  - https://algorithmsinsight.wordpress.com/graph-theory-2/a-star-in-general/implementing-a-star-to-solve-n-puzzle/

Heuristic Function

- -> Manhattan Distance
  - https://medium.com/analytics-vidhya/euclidean-and-manhattan-distance-metrics-in-machine-learning-a5942a8c9f2f#:~:text=Manhattan%20distance%20is%20a%20metric,%2Dcoordinates%20and%20y%2Dcoordinates.
- -> Hamming Distance
  - https://www.tutorialspoint.com/what-is-hamming-distance
- -> Euclidean Distance
  - https://www.cuemath.com/euclidean-distance-formula/
- -> linear conflict
  - https://medium.com/swlh/looking-into-k-puzzle-heuristics- 6189318eaca2

Programming Language

- -> Python
  - https://docs.python.org/3/

Libraries

- -> tkinter
  - https://docs.python.org/3/library/tkinter.htm

THANK YOU

ANY QUESTION