CE368

4. Project Narrative

4.1 Description:

Project Title: Using A* Search Algorithm to Solve N-Puzzle Game.

N-Puzzle Problem: Search strategies rely on heuristics techniques to find the shortest path from an initial given state to a final given state (The Goal State). Understanding these search strategies and the heuristics techniques that they employ is of great importance to AI research. N-Puzzle, also known as the sliding tile puzzle, consists of a grid of sliding tiles that are numbered from 1 to N where the (N+1)th tile is kept empty. The objective of the puzzle is to slide the tiles from an initial state to a goal state. Tiles in the same row or column of any state can be moved by sliding them horizontally or vertically, respectively. Figure 1 is an example of an N-Puzzle with N=8.

Initial State:

	2	3
1	4	5
8	7	6

Final/Goal State:

1	2	3
8		4
7	6	5

Figure 1

Search Algorithm: N-Puzzle supports five different Search Algorithms. These search algorithms can be classified as Informed algorithms or Uniformed algorithms.
N-Puzzle supports different Uninformed Search Algorithms such as:

- Breadth-first Search
- Depth-first Search
- Iterative Deepening Search

N-Puzzle supports different Informed Search Algorithms such as:

- A* Search
- Greedy Search

The A* algorithm is a graph traversal and path finding algorithm. It finds the shortest path between a starting node and a goal node in a weighted graph. The A* algorithm uses three key components: g(n), h(n), and f(n) with f(n) = g(n) + h(n). The path cost function g(n) represents the exact, known distance from the initial starting node to the current position in our search. The heuristic function h(n) provides an estimated cost from the current node to the goal node.

If an Informed Search Algorithm was chosen, then you will also need to select a Heuristic Function.
Heuristic Function: N-Puzzle supports different Heuristic Functions such as:

- Euclidean Distance
- Manhattan Distance
- Tiles Out-of-place

4.2 Technical requirement and Pseudocode:

Problem Requirements:

1. Use the A* Search as the Informed Search Algorithm:
   a. **Nodes**: Points in your graph, **Edges**: Connections between nodes.
   b. **Path Cost**: The actual cost of moving from one node to another.
   c. **Heuristic**: An estimated cost from any node to the goal.
   d. **Search Space**: The collection of all possible paths to explore.
2. Use 0 to represent the empty space
3. Use the following two methods for the calculation of $\boldsymbol{h(n)}$
   a. Use Tiles out-of-place method as Heuristic Function: $\boldsymbol{h(n)}$ = number of misplaced tiles between the initial state and the goal state
   b. Use Manhattan Distance as Heuristic Function: For coordinates $(\boldsymbol{x_1}, \boldsymbol{y_1})$ of the current node and $(\boldsymbol{x_2}, \boldsymbol{y_2})$ of the goal node, these distances are calculated as:

$$\boldsymbol{h(n) = |x_1 - x_2| + |y_1 - y_2|}$$

The Pseudocode for N-Puzzle Problem:

1. Set A* algorithm as your Informed Search Algorithm.
2. Set the Heuristic Function to be used.
3. Set the value N.
4. Set the Initial State
5. Set the Goal State
6. Is Initial State Same as Goal State?
   IF yes, then: Go to step 11
   IF No, then: Go to Step 7

7. Current State = Initial State
8. For the Current state:
    a. Find g(n), Find h(n), and Find f(n)
    b. Find EXP. Where, EXP is used to check if the current state needs to be expanded (EXP = 1) or no (EXP =0)
    c. Find GST. Where, GST is used to check if the current state is the Goal State (GST = 1) or no (GST =0)
    d. Find PST. Where, PST is used to check if the current state is one of the nodes on the path between initial and goal states (PST = 1) or no (PST =0)
9. N-Puzzle: Creating and Updating the Search Tree:
    a. Find all States generated from the current state
    b. Any generated state that was previously discovered needs to be discarded and not to be considered.
    c. For all remaining generated states: Find g(n), h(n), and f(n)
    d. Select the state(s) to be expanded.
        i. The selection is done by selecting the state with minimum f(n).
        ii. In case there is more than 1 state with the same minimum f(n), all these states should be selected.
    e. Find PST, EXP, and GST for each generated state including selected state(s).
    f. Show/display the corresponding search Tree (See Worked Example).
10. Is one of the selected State(s) == goal state?
    IF Yes, then:
        i. Done. Goal State reached.
        ii. Go to step 11
    IF No, then:
        i. For each selected state:
            a. Current State = Selected State
            b. Go back to 8
11. Once the Goal state is Reached:
    a. Find the total numbers of selected states that was needed to reach the goal state.
    b. Find the total number of moves to reach the goal state starting for the initial state.
    c. Show/display the corresponding final Search Tree.

Worked Example:
Find the most effective path to reach the goal state starting from the initial one.

Initial State:

2	8	3
1	6	4
7		5

Final/Goal State:

2	8	3
	1	4
7	6	5

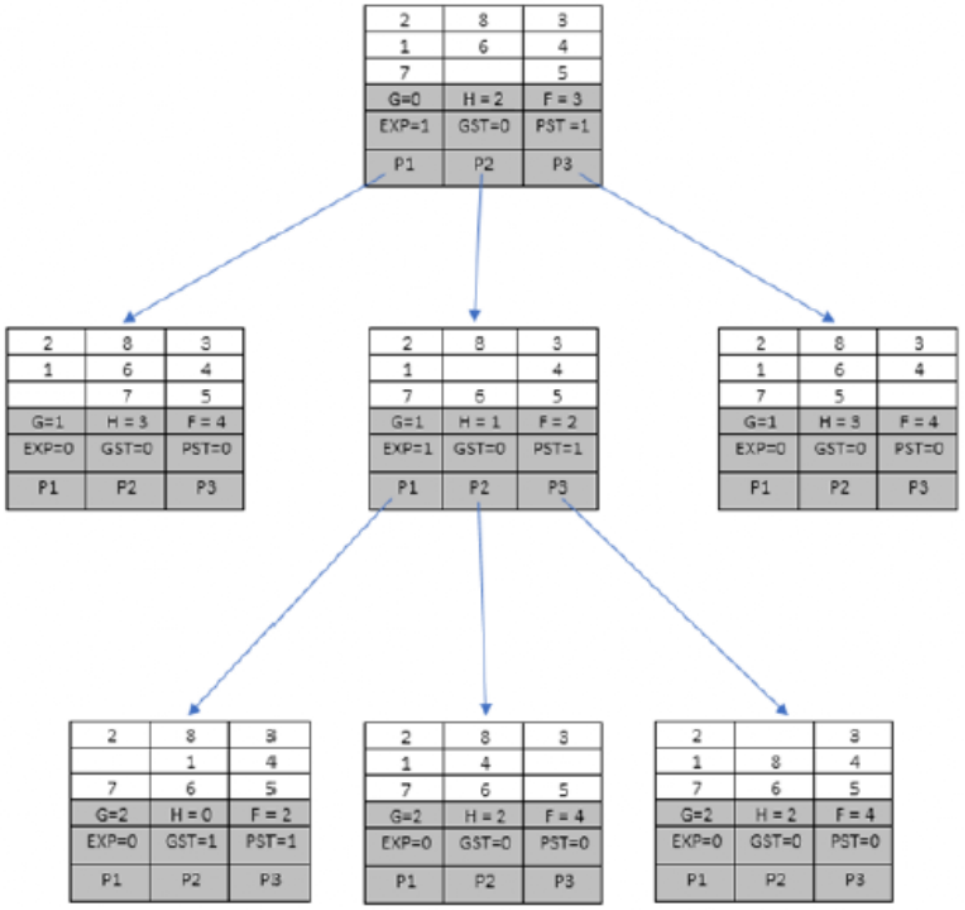
Figure 2

From the initial state, we can conclude that N = 8.
Therefore, the node structure to be used is shown in the table below:

D11	D12	D13
D21	D22	D23
D31	D32	D33
g	h	f
EXP	GST	PST
P1	P2	P3

Table 1

Using the Tiles out-of-place method, the Corresponding Final Search Tree will be as follows:



The C program implements the tree nodes data structures to build the tree. The node structure to be used is as follows:

```
struct TreeNode {
        int D11;
        int D12;
          :
          :
        int DV(N+1)V(N+1) ;
        int EXP;
        int GST;
        int PST;
        int g;
        int h;
        int f;
        struct TreeNode* P1 ;
          :
        struct TreeNode* PV(N+1) ;
};
```

And can be represented as follows:

D_{11}	D_{12}	.....	$D_{V(N+1)V(N+1)}$	g	h	f	EXP	GST	PST	P1	....	$P_{V(N+1)}$

Where: Dij represents the value for a tile at position (x_i, y_j). Dij is equal 0 to represent the empty tile. EXP is used to check if the current state needs to be expanded (EXP = 1) or no (EXP =0). GST is used to check if the current state is the Goal State (GST = 1) or no (GST =0). PST is used to check if the current state is one of the nodes on the path between initial and goal states (PST = 1) or no (PST =0). Finally, g represents g(n), h represents h(n), and f represents f(n),

The program must adhere to the project requirements below:

- The C program should work correctly without any error.
- Functions should take into consideration all special cases.
- The program should be well documented and proper comments must be provided.
- The program should not fail for a typical input and minor special cases (proper testing should be done before submitting the full code).
- Students should be prepared for answering any of the questions related to all that have been covered in the project (this includes both theoretical and practical parts).

4.3 Deliverables:

- **Deliverable 1**
  - Do not use ChatGPT.

- A Power Point Presentation (PPT) with voice over must be prepared and submitted (Recording & Adding your Audio to slides in your PowerPoint. All Students should participate. Do not exceed 4 minutes/student). The PPT should include the following:
  - Using the above pseudocode and explanation (from section 4.2), draw a flowchart for the N-Puzzle algorithm along with a description of the different stages you have.

  Using Tiles out-of-place method as the Heuristic Function, Answer the below:
  - Test the algorithm for scenarios 1 and 2 in appendix A. Show and explain the results and conclude. Solve this question by hand and Do not write a C Code. The C code is only required for Deliverable 2.
  - Show the corresponding tables with all entries (As Table 1) for scenarios 1 and 2 in appendix A by showing all the calculation steps in details. Explain the results. Solve this question by hand and Do not write a C Code. The C code is only required for Deliverable 2.
  - Show the intermediate and final search trees built for scenarios 1 and 2 in appendix A.
  - Find the height for the final search tree for scenarios 1 and 2 in appendix A. Show and explain the results and conclude. Solve this question by hand and Do not write a C Code. The C code is only required for Deliverable 2.

  Using Manhattan Distance method as the Heuristic Function, Answer the below:
  - Test the algorithm for scenarios 3 and 4 in appendix A. Show and explain the results and conclude. Solve this question by hand and Do not write a C Code. The C code is only required for Deliverable 2.
  - Show the corresponding tables with all entries (As Table 1) for scenarios 3 and 4 by showing all the calculation steps in details. Explain the results. Solve this question by hand and Do not write a C Code. The C code is only required for Deliverable 2.
  - Show the intermediate and final search trees built for scenarios 3 and 4 in appendix A.
  - Find the height for the final search tree for scenarios 3 and 4 in appendix A. Show and explain the results and conclude. Solve this question by hand and Do not write a C Code. The C code is only required for Deliverable 2.

- Deliverable 2 - (100 points) 15% (Week14):

  In Deliverable 2:
  - Students need to form project groups (Same Group of students as in deliverable 1).
  - In Deliverable two, a scientific report must be written and an interview will be held.
  - Do not try to copy exact code from the websites. Do not try to copy from other groups. Do not use ChatGPT

  - Scientific Report - Week 14 - [7.5%]:
    - You have to answer the following:
      - Write a C code to implement the N-puzzle algorithm. The program must be menu driven. Upon running the program, the user is given a menu with the functions that can be performed.
        - The user should be able to find the goal state from any given initial state with N = 8 or 15

- The user should be able to select any heuristic function to find the goal state.

- Test your code for all scenarios in **appendix A**. Compare results from PD1 with results from PD2 and conclude.
- Show/Display the corresponding tables **with all entries** (As Table 1) for each scenario in **appendix A** by showing all the calculation steps. Show the result and conclude.
- Show/Display **the intermediate and final search trees** for each scenario in **appendix A**.

Using **Manhattan Distance** method **as the Heuristic Function, Answer the below:**

- Test your code for all scenarios in **appendix B**.
- Show/Display the corresponding tables **with all entries** (As Table 1) for each scenario in **appendix B** by showing all the calculation steps. Show the result and conclude.
- Show/Display **the intermediate and final search trees** for each scenario in **appendix B**.
- Find the height for the final search tree for each scenario in **appendix B**.

The scientific report must be **written** and submitted **through Moodle** and should include the following:

- Full analysis of the above questions
- The written C code along with comments must be included in the report. (**Source code and comments** to describe your code).
- Screen shots of the all outputs must be provided in the report. The screen shots should demonstrate the functionality of the program showing all menus and showing the correct execution of all the functions.