IDA* Search Implementation for the 15-puzzle and 25-puzzle Sliding Problem

Group Members:

Jinyoung (Clara) Hong

Siddharth Gupta

NOTE:

> We worked on this project using the **pair-programming** methodology. In this we basically sat down and one person would code and the other would be actively reviewing the code currently written. We would switch the roles of **programmer** and **reviewer** every time we got together to work on the project.

Project Overview:

- We were required to implement a **memory-efficient** variant of the `A* Search Algorithm` .
- We chose the `IDA* Search` algorithm.
- The implementation is in `C++` .
- We chose to incoporate one *heuristic* in our implementation: `The Manhattan Distance Heuristic` .
- We use a pattern database for the manhattan distance heuristic.

Summary:

- We initialize 20 instances of the `15-puzzle` . These instances are initialized with a **random** board state, which is solvable in nature.
- We then use our implementation(s) to solve and get certain stats about solving each of these instances. Stats are namely:
  - `Time` to solve.
  - `Nodes visited` before reaching goal.
  - `Cost of shortest path` to reach goal.
- We also made a solver for the **25 puzzle** problem using the `IDA* Search` algorithm:
  - This solver is not the most optimal but it does solve a “good-enough” instance of a random 25 puzzle.
  - It uses the manhattan distance heuristic as well.

Running Instructions:

File directory tree is as follows:

```
.
├── 15p.cpp
├── 25p.cpp
├── report.xlsx
├── Makefile
└── REPORT.md
```

- We made 2 separate files for each solver:
  - **`15p.cpp`** => The 15 puzzle solver.
  - **`25p.cpp`** => The 25 puzzle solver.
- You can run the desired program using the following commands (once in the root directory of the project in your terminal)[On an Ubunutu machine]:
  - **`make p15 && ./a.out`** => For running the **15-puzzle solver**.
  - **`make p25 && ./a.out`** => For running the **25-puzzle solver**.

Implementations(s) Summary:

Most of these functions are standard and are present in both the source code files **`15p.cpp`** and **`25.cpp`** :

- **`MakeMovableTable()`** => Return the possible moves each tile can has.
- **`MakeMDTable()`** => Pattern database for manhattan distance heuristic.
- **`GetDistance()`** => Calculate distance for each tile.
- **`GetManhattan()`** => Return value of manhattan distance as a character
- **`GetBlank()`** => Return index of the blank tile.
- **`PrintPath()`** => Print the path taken.
- **`PrintPuzzle()`** => Print the puzzle in a nicely foramtted manner.
- **`IdaStar()`** => `search()` implementation from the pseudo-code of IDA*.
- **`IDAStar()`** => `ida_star()` implementation from the pseudo-code of IDA*.
- **`ShuffleArray()`** => Shuffle a given array (**`state`**).
- **`solvable()`** => Checking the solvability of the current state.

Citations:-

- IDA* Pseudocode - **[https://en.wikipedia.org/wiki/Iterative_deepening_A (https://en.wikipedia.org/wiki/Iterative_deepening_A)*]**
- ShuffleArray() function - **[https://www.geeksforgeeks.org/shuffle-an-array-using-stl-in-c/ (https://www.geeksforgeeks.org/shuffle-an-array-using-stl-in-c/)]**
- C++ reference for various header modules used - **[http://www.cplusplus.com/ (http://www.cplusplus.com/)]** and **[http://cppreference.com/ (http://cppreference.com/)]**