Assignment 1

Submitted by: nbafna, nrpai, sjejurka

Part 1: 15 Puzzle problem

Introduction:

The **15-puzzle** also called **Game of Fifteen** is a sliding puzzle that consists of a frame of 15 numbered square tiles in random order with one tile missing. It is a 4×4 tiles board. The objective of the puzzle is to place the tiles in order by making sliding moves that use the empty space.

In this assignment, we are implementing the classic puzzle with 3 variants:

1) **ORIGINAL:**
   This allows the tile to slide into adjacent empty space using directions UP, DOWN. LEFT, RIGHT defined by valid indexes.

2) **CIRCULAR:**
   This is extension of the Original move, along with the wraps if the empty tile is on the edge of the board. So the tile on the other side of the board can be moved to empty space.

3) **LUDDY:**
   In this all moved are in shape of L, which allows empty tile to be filled with tile that is two positions to the left or right and one position up or down, or two positions up or down and one position left or right.

Implementation:

The algorithm used to solve this puzzle is A*. The search problem consists of:

1) One initial state.
2) A set of legal actions. Actions are represented by operators or moves applied to each state. The operators or moves are defined by the variants as explained above.
3) A goal state.

The A* algorithm is the search implemented with the following function:

$$f(n) = g(n) + h(n)$$ where,

g(n) - measures the length of the path from any state n to the start state

h(n) - is the heuristic measure from the state n to the goal state.

Abstractions:

The abstraction of this problems are defined as:

- **Sample space:**

  All possible permutations of the numbers of the 4 x 4 board.

- **Initial state:**

  It is the board given as input at start of the program.

- **Edge Cost:**

  The edge cost in this case is 1. As each transition is one move.

- **Successor function:**

  The successor function returns all the possible moves possible from the current state.
  To get the next state, the moves are defined based on our variants Original, Circular, Luddy.
  The successor function gives all such possible next states.

- **Goal Sate:**

  We have only one goal state defined as:
  [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0]

- **Strategy:**

  The starter code was implemented using BFS algorithm. Although, it did gave optimal solution its performance was not good for boards that needed more than 10 moves as it is a blind search and it had no picture of its next states and only bothered if the state was goal state or not. To increase the performance we implemented this using A* algorithm. We evaluated it using below heuristics:

Heuristics:

1) Manhattan distance and Linear conflict:

We implemented, the combination of Manhattan distance and linear conflict which greatly increased the performance of boards that were solved in more than 40 moves. However this heuristic was admissible only for Circular and Original moves as the tile changed its position by one move.

- ✓ ADMISSIBLE
- ✓ CONSISTENT

2) Number of misplaced tiles:

We implemented this heuristic, as Manhattan distance was not admissible in L moves calculation. Misplaced tiles gave correct information of the states and greatly improved the performance of the search. This heuristic is simply calculated the count of number of misplaced tiles in the board.

- ✓ ADMISSIBLE
- ✓ CONSISTENT