CSE 410 - AI: Homework 1

This homework will have, both a short written and coding assignment. The problems that are supposed to be written are clearly marked.

1) **(10) (Written) Defining a search problem** What is one possible definition of states, actions, and costs for the "sliding numbers" puzzle. The board is a $n \times n$ with one free square.

2) **(20) (Written) Analyzing search on paper.** For the $n \times n$ sliding number s problem please give estimates of:

- Number of states
- Branching factor
- What is the largest n you expect to be able to solve using a graph based search?

3) **(70) Implementing Search.** In this problem you will implement various searches on a $3x3$ board. In it the search problem is defined with a board state that is the position of all the tiles and the empty spot and four possible actions of "'moving"' the empty square up, left, down, or right. For this you can use the helper functions and classes provided in "'SlideProblem.py". It defines:

- Class `problem` with members:
  - a state `goalState`
  - a state `initialState`
  - function `goalTest(s)` that returns a boolean
  - function `apply(s,a)` that returns a state
  - function `applicable(s)` that returns a list of possible actions
- Class `state` which defines the state of the board
- Class `node` that you can use to build a search graph.
- Helper functions that operate on nodes: `childNode(n,a,p)` and `solution(n)`.

Please use these definition to make implement the class 'Searches' with two member function that define graph-based BFS (graphBFS), and iteratively deepening depth first serach (idDFS), which is a tree based method. Skeletons for these functions as well as usage examples for the helper functions are provided in 'Searches.py'

- **(35)** Breadth first graph search. It should take a problem and be called `graphBFS(p)` and return a solution if there is one or return nothing.
- **(35)** Iteratively deepening, depth limited DFS called `idDFS(p)`

See `Searches.py`, which you can use as a template for the call structure. Note, the solution depth for test cases will be up to 10. Please use these names for the search functions and submit your `Searches.py`.