CS4750/7750 HW #2 (20 points)

Fall 2018

In this programming assignment, follow the Tree-Search and Graph-Search pseudocode in the lecture slides (copied below) and the selected search strategies as presented in the lecture slides to implement the following three search algorithms to solve the 15-puzzle problem:

a) iterative deepening tree search (IDS),
b) depth-first graph search (DFGS), and
c) A* tree search with the total Manhattan distance heuristic.

During search, a tie between search nodes is broken based on the increasing order of index of the tile to be moved.

```
function Tree-Search(problem, fringe) returns a solution, or failure
    fringe = Insert(Make-Node(Initial-State[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node = Remove-Front(fringe)
        if Goal-Test(problem,State(node)) then return node
        fringe = InsertAll(Expand(node, problem), fringe)
    end

function Graph-Search(problem, fringe) returns a solution, or failure
    closed = an empty set
    fringe = Insert(Make-Node(Initial-State[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node = Remove-Front(fringe)
        if Goal-Test(problem,State[node]) then return node
        if State[node] is not in closed then
            add State[node] to closed
            fringe = InsertAll(Expand(node, problem), fringe)
    end
```

Run your algorithms on the following two test cases:

(1)

```
01 02 07 03
05 06 11 04
09 10 15 08
```

```
13 14 12
```

(2)

```
05 01 07 03
09 02 11 04
13 06 15 08
   10 14 12
```

The goal configuration is

```
01 02 03 04
05 06 07 08
09 10 11 12
13 14 15
```

You are allowed to use any programming language in your implementation.

Terminate your program when it expands 1 million search nodes without finding a solution.

Your submission consists of two files:

1) A pdf file containing your team member names, a brief description of your implementations, the software and hardware used in the experiments, and the results on the two test cases, which include the following for each algorithm on each test case:
    a. Report the states of the first 5 search nodes in the order they would be expanded.
    b. Report the solution found (i.e., the sequence of moves) and the total number of moves in the solution.
    c. Report the number of nodes expanded.
    d. Report the CPU execution time in milliseconds.

2) A zip file containing your code with appropriate comments. You may use code found on the Internet, but need to give appropriate credits.

You may form teams of up to 3 people. Each team needs one submission on Canvas.