Homework #1

Due Datess: Task1: 09/28/20, 9PM; Task2: 10/7/2020, 9PM

Submit the zip files on Canvas by the deadlines

1. **Programming Assignment**: The goal of this assignment is to develop a program, in JAVA, C++, or PYTHON, that implements search algorithms for solving a seventeen-tile puzzle (6 rows X 3 columns). The search-based solution must follow the strategy of using a priority queue for the **Open List** and another data structure of your choice for the **Closed List**. The main steps of the search algorithm must be the same as described in class, that is, a repetition of the following steps until either a path to the Goal State is discovered, or you determine that enough effort has been spent a path to thee goal node may not exist.
   1. Remove the highest priority state *S* from the Open List
   2. Check to see if this state *S* is the goal state. If *S* is the goal state then output the path from the start state to the goal state, print the relevant information of the solution as outlined below, and stop the search algorithm.
   3. (If *S* is not the goal state, then) Find all successor states of the state *S*, compute their priority values, and insert them into the Open List. Place the state *S* on the Closed List. If the board description contained in a newly generated search state already exists in a state on the Closed list then do not add this state to the Open list. If the board description included in a newly generated state already exists on the Open list, then keep both the states on the Open list.

Each state should be a structure (object?) containing the following data fields:

* State ID (An integer Number)
* State ID of the Parent State (From which this state was generated)
* A 1X18 vector describing the current 17-tile puzzle board. For example, the board shown here translates to the vector representation shown below:

|  |  |  |
| --- | --- | --- |
| 5 | 19 | 7 |
| 9 | 11 | 10 |
| 14 | 6 | 16 |
| 12 |  | 2 |
| 4 | 15 | 1 |
| 3 | 8 | 13 |

[5 19 7 9 11 10 14 6 16 16 12 **0**  2 4 15 1 3 8 13]

Notice that the blank slot on the board is represented by a ‘**0**’ in the State-vector.

* The g(n) value for the state.
* The h(n) value for the state
* The f(n) (= g(n)+h(n)) value of the state.
* Priority value assigned to this state by your algorithm. It could be the level number in the search tree, g(n), h(n), or f(n), or any other value depending on the search algorithm being used.

The characteristics of your program must include the following:

1. The program takes as inputs two vectors, the first one describing a **start state** and the second one describing a **goal state**. For example, the input to the program may look like:

[5 15 7 8 9 11 10 3 12 0 2 13 4 14 1 6 16 17] [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 17 16]

1. The output of the program must include the following items:
   * 1. **TASK-1:** BFS Algorithm:
        1. A sequence of State descriptions in which the first state is the start (or the goal) state and the last state is the goal (or the start) state as obtained by the Breadth First Search (BFS) Algorithm. Show the details of each state description on the path, including their ID’s, and level number values.
        2. The count of nodes added to the Open List during the search process; and the count of nodes added to the Closed List during the search process (for BFS Search).
     2. **TASK-2:** A\* algorithm using h1 heuristic (described in (d) below):
        1. A sequence of State descriptions in which the first state is the start (or the goal) state and the last state is the goal (or the start) state as obtained by the A\* algorithm using the h1 heuristic (described below). Show the details of each state description including the ID’s, g(n), h(n) and f(n) values.
        2. Length of the path between the start and the goal state in terms of the number of moves; the count of nodes added to the Open List during the search process; and the count of nodes added to the Closed List during the search process (A\* Search using heuristic h1).
     3. **TASK-3**: A\* algorithm using h2 heuristic (described in (e) below):
        1. A sequence of State descriptions in which the first state is the start (or the goal) state and the last state is the goal (or the start) state as obtained by the A\* algorithm using the h2 heuristic (described below). Show the details of each state description including the ID’s, g(n), h(n) and f(n) values.
        2. Length of the path between the start and the goal state in terms of the number of moves; the count of nodes added to the Open List during the search process; and the count of nodes added to the Closed List during the search process (A\* Search using heuristic h2).
2. The cost of making a moving a tile, g(n), in this game is as follows: Whenever a tile numbered 1 through 6 is moved the cost of making this move is 1 unit. Whenever a tile numbered 7 through 16 is moved the cost of making this move is 3 units. Whenever the tile numbered 17 is moved the cost of making this move is 15 units.
3. The heuristic function h1 for the A\* search is as follows: The estimated cost of taking a board B to the goal state G is the number of tiles in B that are not in the correct location as required by G.
4. The heuristic function h2 for the A\* search is as follows: The estimated cost of taking a board B to the goal state G is the **sum** of the smallest number of moves for each tile, that is not at its final location, to reach its final location as required by G.
5. Make sure that your program has the provision to suspend the search process if a solution has not been found after significant resources (time or memory) have been exhausted.

The following items, bundled in a single zip file, must be submitted on Blackboard in response to this assignment:

1. Source code file for your program, named <Your First Initial><Your Middle Initial><Your Last Initial>’SourceCodeFile’.\_\_\_ Your program must be written in one of these three languages: JAVA, PYTHON, or C++.
2. A README.pdf file listing the command line instructions needed for (i) compiling your program, and (ii) executing your program with two 1X20 vectors as inputs.
3. Output of your program obtained for the following two cases, included in a file named: <FI><MI><LI>.OutPutFile.pdf:
   * 1. [1 13 3 5 6 9 11 **0** 8 12 14 10 7 16 15 4 17 2] [1 13 3 5 6 9 11 14 8 12 16 10 7 17 15 **0** 4 2]
     2. [1 13 3 5 17 9 11 **0** 8 12 14 10 7 16 15 4 6 2] [5 1 3 13 17 9 11 0 8 12 14 10 7 16 15 4 6 2]

**Due Date(s):** Task-1: Sept. 28th, 2020, 9PM

Task-2: Oct. 7th, 2020, 9PM

**Grading of the project** will be out 50 points for Task-1 and out of 50. Points for Task-2. For each task the distribution will be as follows:

1. 40% points for the reasonableness of your submitted program files and outputs.
2. 10% points for the readability of the output file. That is, there should be a text description of what each number is and which input corresponds to which output, etc.
3. 10% points for the comments in the source code that describe the role of each section of the code and their role/contribution in the overall program.
4. 40% points for the results obtained by the execution of your program on input test patterns used by TA during evaluation.