University of Dhaka
Department of Computer Science and Engineering
CSE-4111: Artificial Intelligence Lab
4th Year 2nd Semester, 2020

Programming Assignment 1
by Dr. Muhammad Ibrahim and Dr. Md. Mosaddek Khan

Implementation and Performance Comparison of Classical Search Algorithms Using N-Puzzle Problem

A. Problem Description

N-puzzle is a well-known problem used for a long time by Artificial Intelligence (AI) researchers as a benchmark for comparing the performance of search algorithms. This assignment requires the students to implement various classical search algorithms for solving n-puzzle problem. It also entails performance analysis of these algorithms. The algorithms are: breadth first search (BFS), uniform cost search (UCS), depth limited search (DLS), iterative deepening depth first search (IDS), greedy best first search (GBFS), and A* search.

B. Implementation Details

Several specific conditions and guidelines about the code are given below:

1. Your program will have two options: for testing mode and for offline mode. In the testing mode, the program will prompt the number n (of n-puzzle) as input from user. The program will then prompt the user to input the initial board configuration. Then, the user will be prompted to choose a specific algorithm from a menu, or all algorithms (from the menu). After the algorithm(s) finishes working, the user will be prompted again whether or not he/she wants to continue from the beginning. The offline mode is described in Clause 8.
2. Note that the step cost will be assumed to 1 for all algorithms, and the heuristic for informed search algorithms will be (a) the number of misplaced tiles and (b) the Manhattan distance.
3. The node data structure of a search tree should contain the following information: state (2-d array), parent (node type), possible actions from a state (string: “up”, “down”, “left”, “right”, or equivalent integers), the g_cost of this state (double; to be used in UCS), the depth of this node in the tree (int), the h_cost of this state (double; to be used in GBFS and A*).
4. You should use data structures wisely. For example, oftentimes you need to weigh the pros and cons of list and hashset (e.g., ArrayList vs. HashSet in Java).
5. Your code should be as modular, easily understandable, easily extensible and flexible as possible.

6. In general, you need to follow the pseudo-code of the algorithms given in the textbook (AI: A Modern Approach). However, for the sake of better implementation, minor details of the pseudo-code may be customized or modified.
7. Output of the source code will be redirected to a text file. You should display a lot of useful tracking information as your program runs towards a solution. For example, after adding/removing every node in the frontier and explored sets (if applicable), you may display the number of nodes residing in these sets. You may output which actions of the current state is currently being explored. You may display the intermediate states of the puzzle after performing an action. In a nutshell, you should make the output of your program as easily followable as possible. Finally, after getting the solution, you must display the entire solution path, i.e., display all the actions and intermediate states starting from initial state to the goal state.
8. Apart from console input and output mentioned above, you need to compare the performance of the algorithms on 8-puzzle problem in the following way. You should create -- manually or automatically -- some states of 8-puzzle problem that requires varying steps, such as 1, 2, …, 20, to reach the solution. The performance metrics for our comparison of algorithms are: clock time, the number of nodes generated in the search tree, and the path cost from initial to goal state.

   Since there are three metrics, you will generate three separate plots. Along the x-axis of a plot will be the distance between the initial state and goal state (1, 2, …, 20), and along the y-axis will be a performance metric. That means for six algorithms you will have six curves in each plot.

   These plots may be generated in a single pipeline or by a separate program that takes your output (text) file as input (which must then contain the required numbers such as clock time, the number of nodes generated etc.).

C. Tools and Logistics

Programming language: Java or Python.

Platform: Any (Linux, Windows etc.).

Use of third-party libraries and APIs: One of the goals of this assignment is to develop the students' ability to code from scratch. Hence the use of third-party libraries are not allowed unless approved by the course instructors. The students are asked to contact the course instructors to be clarified should any confusion arises.

D. Submission Details

Some specific information about the task is given below:

Team: No team will be formed. Each student will submit individually.

Report: A typed report will be submitted by each student at the end of the assignment. The report must be between 3 to 5 pages (excluding the title page). Font size will be 11, and normal margin will be used. It is encouraged that the report be produced using Latex typesetting system. The report should contain only the following sections: a brief introduction of the problem, your result analysis, and the challenges you faced during the assignment. You should not write the description or pseudo-code of the algorithms in your report.

Code: Source-code needs to be submitted in a completely ready-to-run form. After the submission of your code, we may not ask for specific instructions from you.
Viva: A viva voce will be conducted during/after the code submission.

E. Marks Distribution

The marks will be given based on the following particulars:
Modularity and extensibility of code: make your code flexible, easier to understand and extend.
Effective use of data structures: choice of data structures may hugely impact your coding effort.
Understandability of output: show as much useful information as possible.
Readability of plots: wisely and academically decide the labels, colors, curves etc.
Report writing: organization and structure, result analysis and interpretation, English language proficiency.
Viva voce: capability to explain the code and modify it on-demand, to explain the problem and algorithms.

F. Plagiarism

While the students can discuss among themselves, each student will obviously accomplish the assignment individually. For both the code and report, plagiarism will be thoroughly checked using plagiarism detection software.

G. Deadline

In three (03) weeks from today.

Further guidelines, if deemed necessary, will be provided in due course.