CS 4346 Artificial Intelligence | Project 2

EVALUaTION OF HEURISTIC SEARCH ALGORITHMS

April 27, 2020

Report Author: Shervone Mayes

Project Contributors:

* Madison Crown (),
* Shervone Mayes (Classes, Driver program, ASTAR\_H1 and ASTAR\_H2 algorithms ) , and
* Erik Pantoja (AStarSN and PatternDatabase algorithms with helper functions)

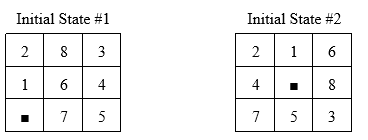
# overview

## Problem Description

|  |  |
| --- | --- |
|  | Explain the problem for which you have developed the various heuristic algorithms |

Group tasked with the problem to implement various heuristic functions. The heuristic functions provide informed search for intelligent systems. The team will implement A\* algorithm and run it for both H1 (Number of Misplace Tiles) and H2 (Manhattan Distance) heuristic functions. Additionally, each team member will develop their own additional heuristic function.

Each heuristic function will have its performance evaluated and analyzed. The results of the performance evaluation will be displayed in a table for the two following initial states:



By comparing various heuristic functions one can observe the advantages and shortcomings associated with each heuristic function implementation and make objective decisions to determine best solution given a specific problem statement.

Heuristic Functions:

* ASTAR (H1)
* ASTAR (H2)
* ASTARSN
* PATTERNDATABASE

## Data Tabulation

|  |  |
| --- | --- |
|  | Tabulate performance data for each implementation of heuristic functions for two initial states defined in project description |

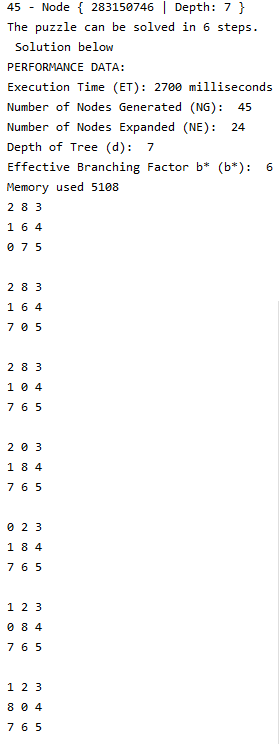
Table Definitions:

* Execution Time (ET)
* Number Of Nodes Generated (NG)
* Number Of Nodes Expanded (NE)
* Depth Of The Tree (D)
* Effective Branching Factor B\* (NG/D)
* Memory Used (MO)

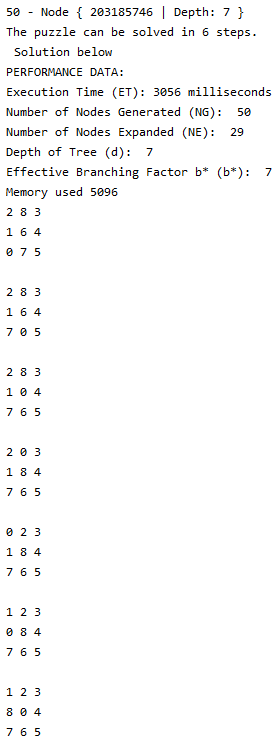
Initial State #1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Heuristic Function | ET (microseconds) | NG | NE | d | b\* | MO |
| ASTAR\_H1 | 2700 | 45 | 24 | 7 | 6 | 5108 |
| ASTAR\_H2 | 3056 | 50 | 29 | 7 | 7 | 5096 |
| ROWSHEURISTIC | 2790 | 60 | 31 | 7 | 8 | 5096 |
| PATTERNDATABASE | 3028 | 50 | 29 | 7 | 7 | 5100 |
| DIAGONAL | 2844 | 22 | 12 | 5 | 4 | 5100 |

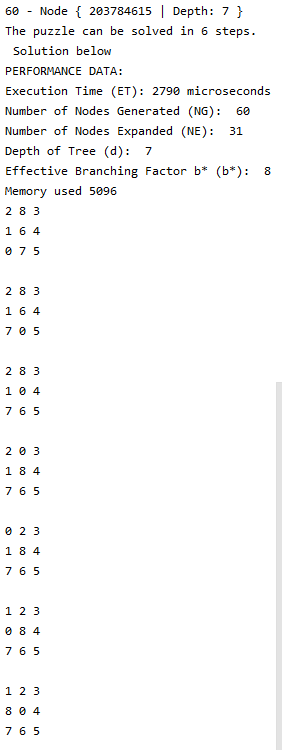
ASTAR\_H1\_I1:



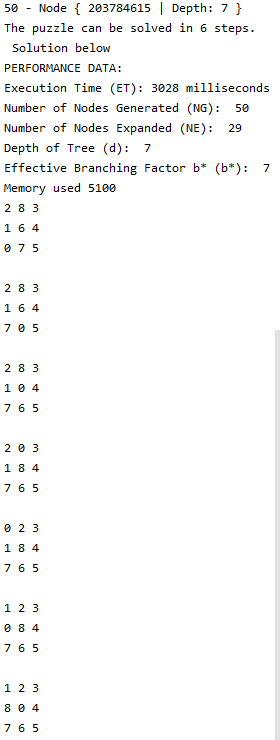
ASTAR\_H2\_I1:



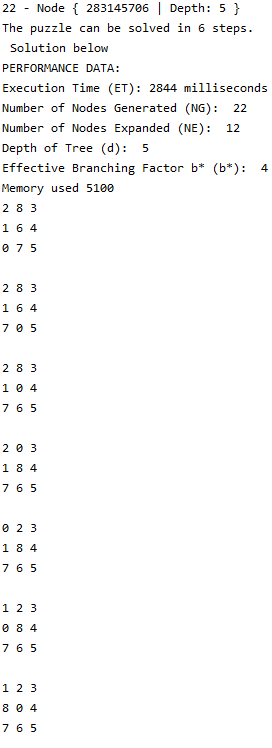
ROWHEURISTICS\_I1:



PATTERNDATABASE\_I1:



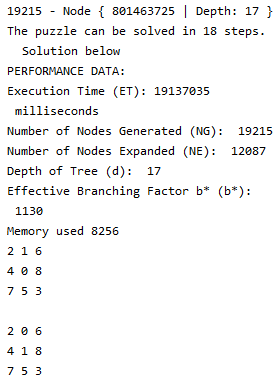
DIAGONAL\_I1:



Initial State #2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Heuristic Function | ET (microseconds) | NG | NE | d | b\* | MO |
| ASTAR\_H1 | 19137035 | 19215 | 12087 | 17 | 1130 | 8256 |
| ASTAR\_H2 | 12423135 | 13982 | 8897 | 16 | 873 | 7548 |
| ROWHEURISTICS | 24023814 | 21260 | 12737 | 19 | 1118 | 8764 |
| PATTERNDATABASE | 380197 | 2317 | 1430 | 19 | 121 | 5480 |
| DIAGONAL | 6580140 | 10041 | 6329 | 16 | 627 | 6704 |

ASTAR\_H1\_I2:

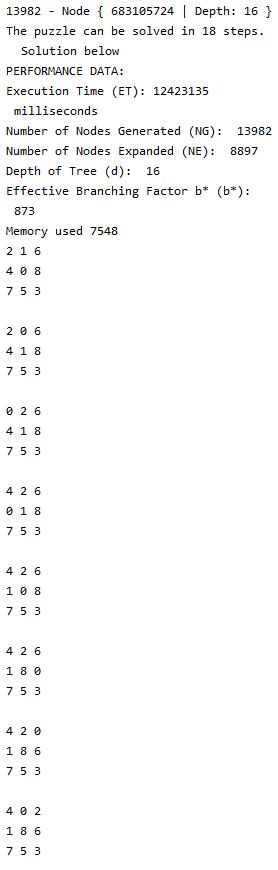


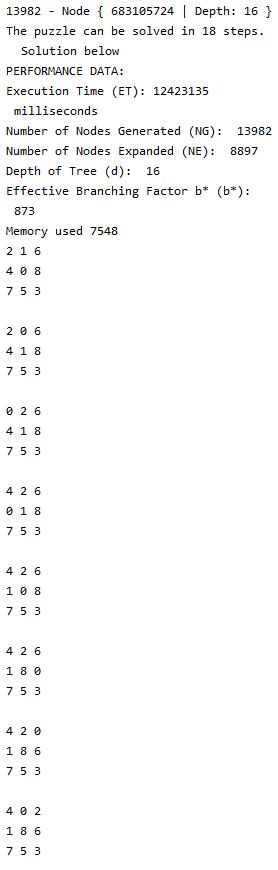
Solution Path:



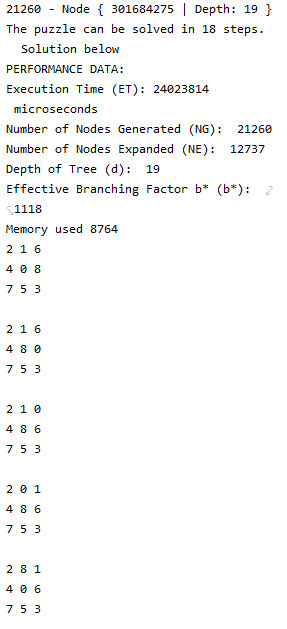


ASTAR\_H2\_I2:

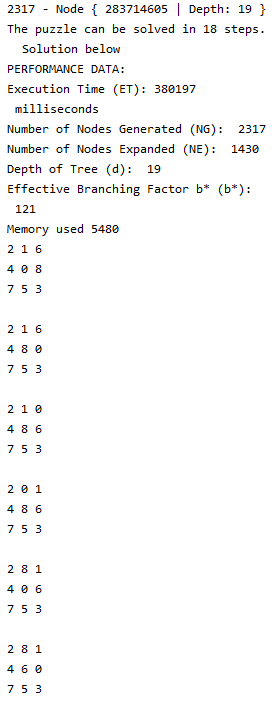


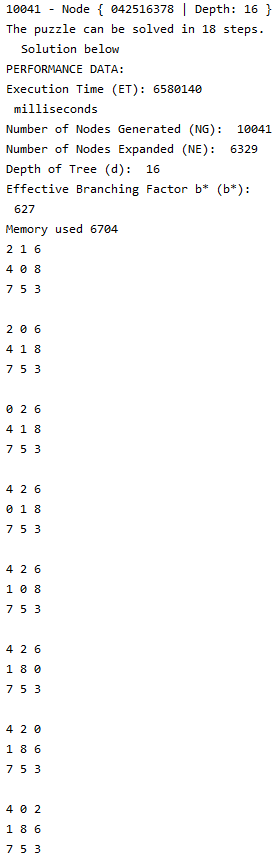
ROWHEURISTICS\_I2:



PATTERNDATABASE\_I2:



DIAGONAL\_I2:

## Data Analysis

|  |  |
| --- | --- |
|  | Analyze the results. That is which evaluation function performed better and why. Write the conclusions you draw from these analyses |

Using JetBrains CLion IDE for code implementation. On Mac OS, CLion provides CPU Profiler Tool using DTrace.

Flame Graph: Provides a visual representation of the execution time; a visualization of profiled software, allowing the most frequent code-paths to be identified quickly and accurately. It is a collection of stack traces. The rectangles stand for frames of the call stack, ordered by width. It is a call tree summary.

How to interpret: Can take a bottom-up or top-down approach. When reading the flame graph, focus on the widest blocks. These blocks are the functions most presented in the profile. You can start from the bottom and move up, following the code flow from parent to child methods, or use the opposite direction to explore the top blocks that show the functions running directly on the CPU.

Call Tree: In the Call Tree tab, you can see a tree of function calls with the corresponding execution time.

Method List:

Context Menu: It allows you to locate the selected function in another tab (for example, Focus on method in Methods List for a Flame Graph block), navigate to the source code (Jump to Source), and copy frame information to clipboard: only the frame name (Copy Frame) or the sequence of frame names from the stack bottom up to the selected frame (Copy Stack up to Frame).

# program

## Program Implementation

|  |  |
| --- | --- |
|  | Explain in detail the program implementations. Also, explain important features you have implemented. |

\*Everything is an object

State Class: Represents the tiles on the puzzle. Represents a unique combination of the tiles. The \_state is represented using a one-dimensional array. The indices represent the location of the tile and the value of the element is the number on the tile. There are getters and setters to get the state. Additionally, there are findEmpty() and swapEmpty() functions for moving the empty tile.

Neighbors Class: Defines neighbors based on where empty tile is. Neighbors generated for each of the nine tiles. The neighbors are stored in an array. The indices represent the location of the potential places empty tile can move. The function getNeighbors() takes in an integer that is the index of the empty tile and returns an array of indices where the empty tile can move. The function CreateGraph8Puzzle() is a map of all the neighbors based on the input value of the empty tile index.

Node Class: Represent each element of tree structure. The Node object \_parent keeps pointer to its parents instead of having a list or an array of pointers to its children. The Node object \_depth defines the depth at which node exists. The Node object \_state represents the state of the 8 puzzle tiles that the node represents. Setter and getter methods implemented to get and set values of the mentioned node objects. Additionally, print() implemented to present node states during program execution.

Solver Class: The

## Source Code

|  |  |
| --- | --- |
|  | Attach a copy of the source code. |

## Program Execution

|  |  |
| --- | --- |
|  | Demonstrate the execution of the program. Print initial state, some intermediate states as they are generated, the final state when it is reached, and finally, the path generated by your program. |

## Program Analysis

|  |  |
| --- | --- |
|  | Explain what features you have added or modified and why. |

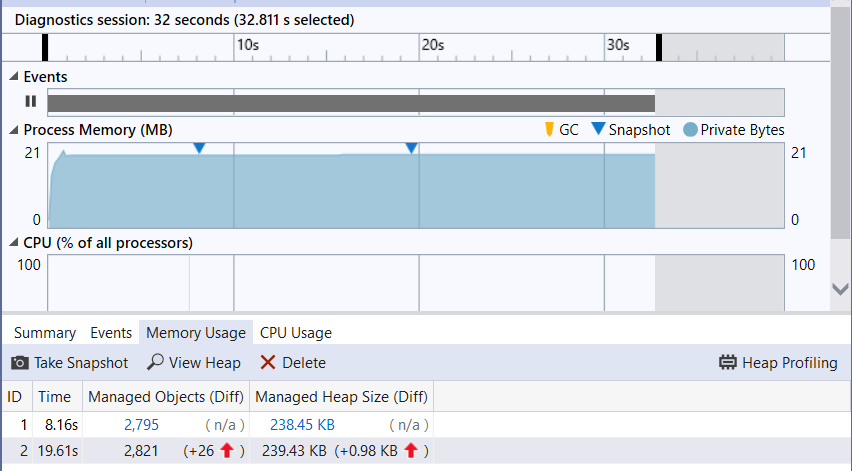
|  |  |  |
| --- | --- | --- |
| **MODE** | **DESCRIPTION** | **JUSTIFICATION** |
| REMOVAL | gets() in original code | but it suffers from Buffer Overflow as gets() doesn’t do any array bound testing. gets() keeps on reading until it sees a newline character.  To avoid Buffer Overflow, fgets() should be used instead of gets() as fgets() makes sure that not more than MAX\_LIMIT characters are read. |
| REMOVAL | go to: statements in original code | Use of **goto** statement is highly discouraged in any programming language because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. Any program that uses a goto can be rewritten to avoid them. |
| MODIFY | from C to C++ | Meet project requirements |
| ADDITION | struct ConclusionListStruct | Ability to combine data of different types as group for similar logical application |
| ADDITION | struct VariableListStruct | Ability to combine data of different types as group for similar logical application |
| MODIFY | from printf to cout | Did not need to return formatted strings |
| ADDITION | Variables for Forward Chaining | In order to keep track of user responses |

## Analysis Results

|  |  |
| --- | --- |
|  | Explain in detail the effect on your results of your program implementations and modified feature. |

Concise. Secure.

Efficiency improvements can be achieved with use of map, dictionary. Opportunity for improvement in future development of this project.



## Conclusion

|  |  |
| --- | --- |
|  | Explain the conclusion and what you learned from the project |

A program of this magnitude requires a significant amount of attention to detail. Users rely heavily on the accuracy of the inputs. As a result, significant testing should be performed. There are a great many resources available online to build such a system at which point some fine-tuning is involved to build a sufficient knowledge base. Garbage In Garbage Out. The system is only as smart as its inputs.

As is always the case working in groups can present both challenges and opportunities. It is best to try to capitalize on the positive attributes of team members and try to best utilize them best on their abilities.

There are opportunities for improvement in the program itself. After further scouring online dictionary implementation may likely improve overall efficiency. As lists grow, the need to main a reasonable time complexity is vital. Though not fully realized here it should be of consideration for future projects to try to aim for optimal solutions.

The hardest part of this project was trying to make an attempt to refactor the existing code. It is significantly easier to start from scratch than try to change existing code. The fact of the matter is however that this is the reality of going into the industry. There may be scenarios in older companies with legacy systems where code needs improving or gets a complete overhaul for a new system implementation, etc. This was a good exercise for such potential scenarios.

## Program Instructions

|  |  |
| --- | --- |
|  | Explain how to run program, download files... |

Program requires three text files for initialization of lists:

* ConclusionList.txt
* VariableList.txt
* ClauseVariableList.txt

These files have been included in electronic submission.

Program execute occurs in main.cpp. Other files are required for proper execution as a result of converting from C to C++. IDE created them autmatically when attempted to compile.

Option to download project by clone from GitLab using the following URL:

* Git Repository: https://gitlab.com/ToborMai/cs\_4346\_ai\_project1.git

# references

## References

|  |  |
| --- | --- |
|  | Add all the references you have consulted in completing the project |

* *Principles of Artificial Intelligenc*e By Nils J. Nilsson
* <https://shawnzhong.com/2019/03/27/analyze-execution-time-with-clion/>
* <https://www.jetbrains.com/help/clion/cpu-profiler.html#>
* http://www.brendangregg.com/flamegraphs.html
* <https://algorithmsinsight.wordpress.com/graph-theory-2/a-star-in-general/implementing-a-star-to-solve-n-puzzle/>
* <https://algorithmsinsight.wordpress.com/graph-theory-2/ida-star-algorithm-in-general/>
* https://www.geeksforgeeks.org/measure-execution-time-function-cpp/