

# Puzzle Sliding Game

## COMP 8700 Fall 2020 Final Project

**Xinmeng Yuan,**

Master of Applied Computing,  
University of Windsor,  
yuan12b@uwindsor.ca

**Ruiting Liu,**

Master of Applied Computing,  
University of Windsor,  
liu72@uwindsor.ca

**Xinyu Ji,**

Master of Applied Computing,  
University of Windsor,  
ji11p@uwindsor.ca

**Mengyao Liao,**

Master of Applied Computing,  
University of Windsor,  
liao71@uwindsor.ca

**Wen Dong,**

Master of Applied Computing,  
University of Windsor,  
dong23@uwindsor.ca

**Abstract** – this is the report of our team final project of Artificial Intelligence Introduction, in this project, we have experimented and explored a set of search algorithms including informed and uninformed ones with the famous Eight Puzzles Game, and found that the performance of different algorithms varies markedly in terms of optimality, completeness, time complexity and space complexity; We also found that heuristic function plays a significant role in the search for informed algorithms, and impact the outcome tremendously.

# Introduction

The project is the final project of our AI Introduction course, which is to put the comprehensive knowledge and theory we have learned from class into reality, from abstract intuition to tangible experiment. It is beneficial for all the team members to consolidate the learning and build hands-on experience on Artificial Intelligence.

In the project, we have chosen and explored a set of informed algorithms:

- AStarSearch – A graph traversal and path search algorithm, an important and widely used search algorithm in computer science due to its completeness, optimality, and optimal efficiency. It uses function  $f(n) = g(n) + h(n)$  as estimate function, based on which to expand cheapest node first. The sole drawback is its space complexity therefore not practical for large-scale problems.
- RecursiveBestFirstSearch – It is a variant version of A Star for large-scale problems where A Star is unable to tackle due to space complexity. Instead of keeping all the frontiers in memory, it uses a flag  $f\_limit$  to keep track of the  $f$ -value of the best alternative path available from any ancestor of the current node.
- GreedyBestFirstSearch – It is designed to find goal quickly by expanding the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly with evaluate function  $f(n) = h(n)$ . So, nodes not on the solution path are not expanded. Cost low, however, it is not optimal neither complete because optimal solution might be in other unexplored paths.

Additionally, two uninformed as well:

- BreadthFirstSearch – Breadth-First is a search strategy in which the nodes are expanded and explored in a breadth first way, that is the root node expanded, then the successors are expanded and so on. When all steps costs are equal, it is optimal as it always expands the shallowest unexpanded node.
- UniformCostSearch – with a simple extension on Breadth-First search, Uniform-Cost Search is optimal for any step-cost function. Instead of expanding the shallowest node, it expands the node with lowest path cost  $g(n)$

Aside from the variety of algorithms, we have also employed three different heuristic functions and two different accessory strategies, i.e. tree search and graph search to work with the informed searches, to observe how and how much they impact the searching process.

## Experiment

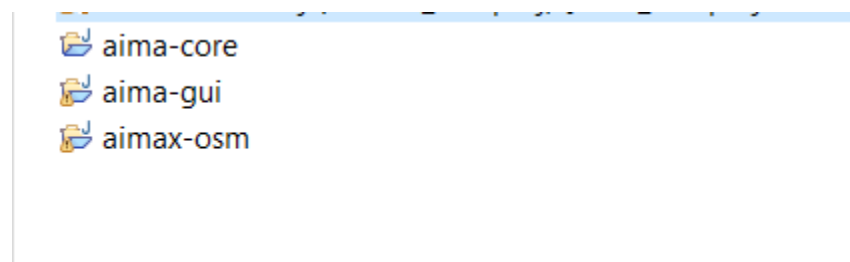
Our experiment is based on the open source project AIMA which is the java implementation of algorithms from Russell and Norvig's Artificial Intelligence – A Modern Approach 3rd Edition.

### Setup

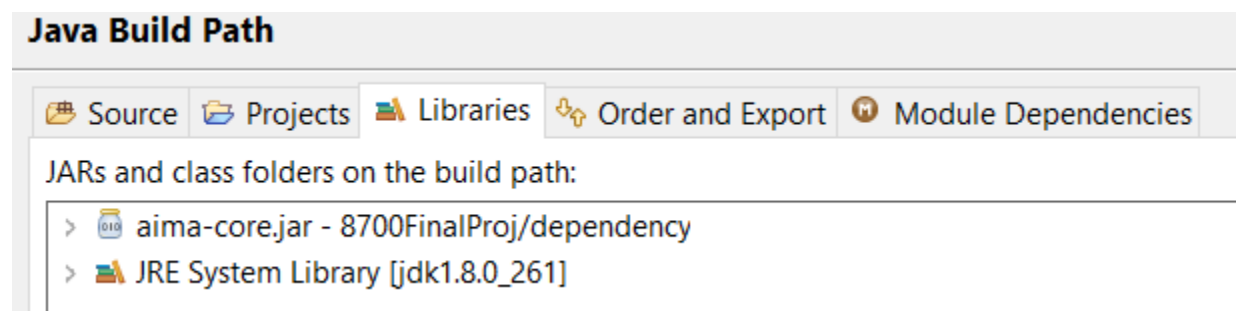
1. Download the latest source code of AIMA java from the GitHub repository

<https://github.com/aimacode/aima-java>

2. open the projects with Eclipse (Version: 2020-09 (4.17.0)) with builder Java 8+



3. Export aima-core as a jar library then add and reference it in our project



4. Finally, we are ready to import all sorts of algorithms and eight puzzles related assets from the library to build our testing flows of interest.

```
12 import aima.core.environment.eightpuzzle.EightPuzzleBoard;
13 import aima.core.environment.eightpuzzle.EightPuzzleFunctionFactory;
14 import aima.core.environment.eightpuzzle.EightPuzzleGoalTest;
15 import aima.core.environment.eightpuzzle.ManhattanHeuristicFunction;
16 import aima.core.environment.eightpuzzle.MisplacedTilleHeuristicFunction;
17 import aima.core.search.framework.SearchAgent;
18 import aima.core.search.framework.SearchForActions;
19 import aima.core.search.framework.problem.Problem;
```

## Coding

Based on the AIMA project, coding the experimental flows is like a breeze, as most of the work can be achieved simply by permutation and combination of various existential Java classes. For instances, testing A Star with graph search and Manhattan heuristic function is implemented as below:

1. Prepare initial board states to search solution for
2. Create an instance of Manhattan heuristic function class

```
ManhattanHeuristicFunction hf = new ManhattanHeuristicFunction();
```

3. Create an instance of A Star search algorithm with the heuristic function and a graph search
4. Assembly the objects together to create a Problem

```
Problem problem = new Problem(initialState,  
    EightPuzzleFunctionFactory.getActionsFunction(),  
    EightPuzzleFunctionFactory.getResultFunction(),  
    new EightPuzzleGoalTest());
```

5. Instantiate a Search Agent to conduct the actual search and output the search result in the end.

```
SearchAgent agent = new SearchAgent(problem, search);  
printInstrumentation(agent.getInstrumentation());
```

Find our project repository below, download and run it if you are interested.

[https://github.com/fanchuanster/8700\\_finalproj.git](https://github.com/fanchuanster/8700_finalproj.git)

## Testing Results and Analysis

### I. Comparison between algorithms

	Algorithm	pathCost	nodesExpanded	queueSize	maxQueueSize	
Informed	A Star	18	227	156	157	Manhattan heuristic function and graph search used. With initial state 5 4 0 3 2 8
	Recursive Best First	18	472607	N/A	18	
	Greedy Best First	58	309	228	229	

						6 1 7
Uninformed	Breadth First	18	13210	7396	7396	
	Uniform Cost	18	19268	11020	11023	

### Result Analysis:

From the result, we can see except Greedy Best First search, all the others are optimal, among which A Star performs well in the number of expanded nodes, and also with a decent queue size and maximum queue size comparing to uninformed approaches. Uninformed ones are also able to find the optimal solution, but with considerable cost as revealed by nodes expanded and queue size.

## II. Comparison between Tree search and Graph search

Algorithm		pathCost	nodesExpanded	queueSize	maxQueueSize
A Star	Graph Search	18	227	156	157
A Star	Tree Search	18	14653	26347	26348
Greedy Best First	Tree Search	Endless searching			
Greedy Best First	Graph Search	58	309	228	229

### Result Analysis:

From the result, we can see Graph search helps reducing the number of expanded nodes noticeably comparing to Tree Search for A Star, and for Greedy Best First search, Graph Search ended with modest cost, while Tree search ran into an endless probe.

## III. Comparison between heuristic functions

Heuristic Function	pathCost	nodesExpanded	queueSize	maxQueueSize
Manhattan	18	227	156	157
MisplacedTille	18	1239	757	758
EuclideanDistance	18	339	230	231

## Result Analysis:

For this experiment, three heuristic functions have been tested as we can see above, two of them were from AIMA library and one was created one our own. although all of them are admissive and consistent and able to identify the optimal solution, the matrix of each search varies. It shows that heuristic function is extremely important for an informed search, and it is crucial to pick an appropriate one which can reflect the real situation between the source state and the destinate goal.

## Interesting found in AIMA

### An AIMA Bug?

During our work, we found many states which turns out to be half of the possible states do not have a solution path towards the goal state, 181440 states. For example:

187432650, 187432605, 187402635

107482635, 017482635, 187042635, ...

And the other half 181440 are resolvable, for example:



436785102, 236154780, 327186450

605237184, 673805142, 140562378, ...

The  $181440 * 2$  happens to be the factorial of the length of the game board size 9, this reminds us there should be something wrong with the Eight Puzzles Game in the AIM library.

### What have we done to overcome the bug?

To proceed with our work smoothly, we distinguished all the unresolvable states from the solvables to ensure valid ones are selected and tested.

 solvable_initial_states.txt	12/19/2020 7:31 PM	Text Document	1,950 KB
 unresolvable.txt	12/15/2020 6:40 PM	Text Document	1,772 KB

Meanwhile, we reported the issue to AIMA Team in hope a fix would be implemented in the future.

<https://github.com/aimacode/aima-java/issues/478>

## Conclusion and Future Work

By exploring different approaches with Eight Puzzle game, we have found that different algorithms perform variously due to the nature of each, in general, informed algorithms excel in performance given a suitable heuristic function. Verified that Tree Search and Graph Search influence the performance matrix in terms of node expanded and queue size during the search. Tested and appreciated that heuristic function plays a significant role in informed searches, and it is crucial to identify a well-suited heuristic function for a well-behaved path finding solution.

In the future, we consider intensifying our work deep into more test states, as well as extending our effort to more search algorithms learned from the class, like Depth First Search, Bidirectional Search, Hill Climbing family, and Genetic Search, ex cetera. We believe that will benefit our learning in the field and consolidate our knowledge gained from the study journey.

## Work distribution among team members

Yuanxin Men – beginning and abstract

Wen Dong – setup and coding

Ruiting Liu, Xinyu Ji – introduction and algorithms selection

Mengyao Liao – testing result analysis and conclusion

## References:

<https://github.com/aimacode/aima-java>    AIMA3e branch

[https://en.wikipedia.org/wiki/A\\*\\_search\\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm)

Artificial Intelligence - A Modern Approach Third Edition