2017

Seyit Yiğit SIZLAYAN-1876861

METU

11/29/2017

EE586 AI - Take Home Exam

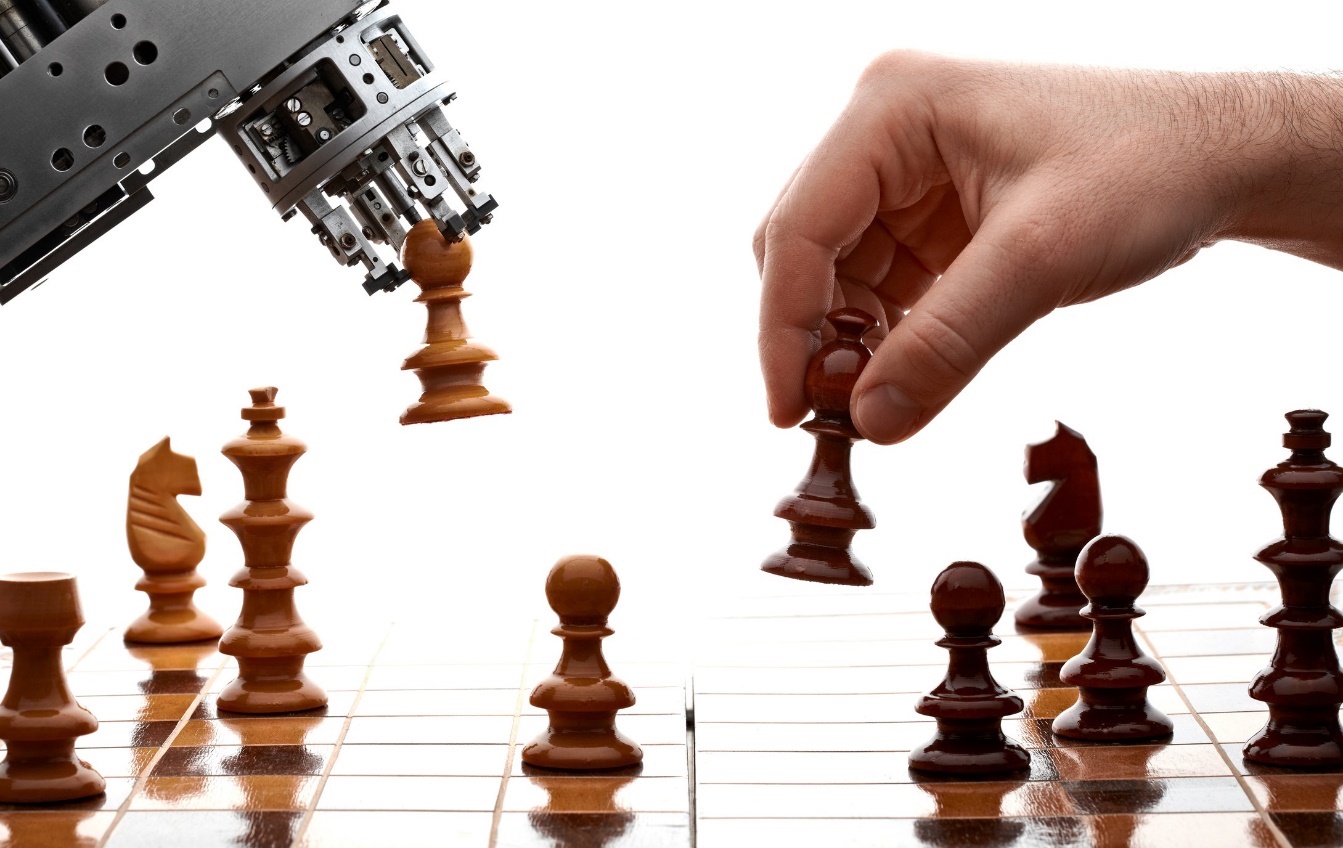


Table of Contents

**No table of contents entries found.**

# Introduction

This project consists of puzzle solving problem with given parameters. For this project, I have chosen Matlab as programming language to be able to generate GUI more easily.

In the program, the instructor wanted us to implement the algorithms given below:

1. Breadth First Search (BFS)
2. Depth First Search (DFS)
3. Iterative Deepening DFS (IDDFS)
4. A\* Heuristic Search

Some algorithms given above has some problem by nature for specific cases. I have added additional functions to overcome these problems which are:

* Memorizing Breadth First Search (BFS\_withMemory)
* Memorizing, Depth Limited DFS (DFS\_limitedDepth)

I will explain them in detail in further parts.

# Data Structures

## Puzzle 🡪 NodeClass

To represent a puzzle, I have turned it to its row version and used “-1” to represent blank. For example

🡪

This row is stored as a NodeClass, which is user defined class, in Matlab. It has properties of:

* BackPointer 🡪 Holds predecessor, required for the map from the bottom
* State 🡪 Holds state of the puzzle
* Depth 🡪 Holds position of from the top, required for DFS, IDDFS and A\*
* HeuristicScore 🡪 Holds the heuristic score for A\*

And methods of:

* Constructor 🡪 Takes a vector and turns it to NodeClass
* successor\_withoutHeuristic 🡪 Takes the current state and returns successors list as NodeClass, heuristic is not calculated
* successor\_withHeuristic 🡪 Takes the current state and returns successors list with calculated heuristic scores with given heuristic function
* heuristicMisplaced 🡪 Misplaced tile number given function
* manhattanDistance 🡪 Manhattan distance for each entry in the puzzle

## Stack

It was required for the DFS, DFS\_limitedDepth and IDDFS implementation. It is simple FILO stack class written in Matlab. It has push and pop functions. Stores the values in an array given type of array.

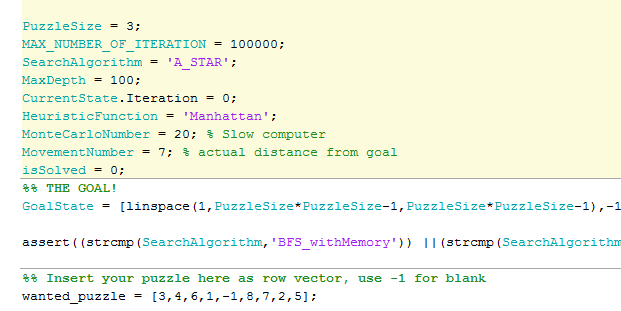
## Queue

It was required for BFS\_withMemory and BFS\_withOutMemory implementation. It is simple FIFO queue class written in Matlab. It has push and pop functions. Stores the values in an array given type of array.

# GUI

Due to unfortunate events happening in last 2 weeks, I was not able to implement beautiful GUI.

However, the wanted properties for the project still can be determined using “globals.m” formatted file.



As you can see from the file snapshot given above, one can determine:

* Puzzle size by changing PuzzleSize variable
* Algorithm by changing SearchAlgorithm variable
  + Possible Values
    - BFS\_withMemory
    - DFS\_limitedDepth
    - A\_STAR
    - IDDFS

Since other algorithms takes too much time to solve, I have limited the functions as this. But:

For standard BFS, one can use:

>> globals

>> [ map, elapsed\_time, visitedNodeNumber, totalNodes ] = BFS\_withoutMemory(initial\_node,heuristic);

>> displayMap(map, elapsed\_time, totalNodes);

For standard DFS, one can use:

>> globals

>> [ map, elapsed\_time, visitedNodeNumber, totalNodes ] = DFS (initial\_node,heuristic);

>> displayMap(map, elapsed\_time, totalNodes);

* Heuristic function by changing HeuristiFunction variable
  + Possible Values
    - Misplaced
    - Manhattan
* Maximum depth for DFS\_limitedDepth by changing MaxDepth variable

For Monte Carlo Simulations:

* Total number of Monte Carlo simulation for specific distance from goal state by changing MonteCarloNumber variable
* Maximum number of movements from goal state by changing MovementNumber variable