# Report Assignment 1

**Search**

**Guillermo Martín-Coello Juarez & Daniel Varela Sánchez**

## Section 1 (1 point)

* 1. **Personal comment on the approach and decisions of the proposed solution (0.5pt)**

This first question consisted of working on a depth first algorithm. To save the path from the parents to the goal node we decided to base our code in the use of a dictionary with all de antecessors of each node.

* + 1. **List & explanation of the framework functions used**

To complete the depth first algorithm we used the provided framework functions. This consisted on:

* getStartState(): function to obtain the state the game begins in.
* isGoalState(state): function with a state as an argument. Returns boolean, true if the provided state is a goal state, else false.
* getSuccessors(state): function with a state as an argument. Returns a list with all the successors of the provided state.

The class Stack from the utils package was also used with the corresponding

methods (push and pop)

* + 1. **Includes code written by students**

|  |
| --- |
| Code for exercise 1 |

* + 1. **Screenshots of executions and test carried out analyzing the results**

|  |
| --- |
|  |
| Successful execution of tiniMazeSearch to prove that the search methods are working as expected |

|  |
| --- |
|  |
| DFS successful on the tinyMaze |

|  |
| --- |
|  |
| DFS successful on the mediumMaze |

|  |
| --- |
|  |
| DFS successful on the bigMaze |

|  |
| --- |
| 7/7 grade in the q1 tester provided |

* 1. **Conclusions on the behavior of pacman, it is optimal (y / n), reaches the solution (y / n), nodes that it expands, etc (0.5pt)**

The result is not optimal (there are better possible paths than the ones achieved).

The result reaches the solution.

The result expands all the nodes needed on a depth first algorithm, meaning it expands first the children.

* + 1. **Answer to question 1.1**

The exploration order is not what was expected. It is not a pacman exploration function, it is an algorithm exploration function. Pacman does not actually go to all the explored squares, but these were explored by the algorithm to find the path to the goal.

* + 1. **Answer to question 1.2**

This is not a least cost solution. Depth first algorithm will not always find the least cost solution. This is because the algorithm always looks for the first path, not the best.

* + 1. **Answer to question 2**

## Section 2 (1 point)

* 1. **Personal comment on the approach and decisions of the proposed solution (0.5pt)**

To accomplish this question we used the skeleton of the previous question and change the mechanism of the algorithm by using a queue instead of a stack.

* + 1. **List & explanation of the framework functions used**

To complete the breath first algorithm we used the provided framework functions. This consisted on:

* getStartState(): function to obtain the state the game begins in.
* isGoalState(state): function with a state as an argument. Returns boolean, true if the provided state is a goal state, else false.
* getSuccessors(state): function with a state as an argument. Returns a list with all the successors of the provided state.

The class Queue from the utils package was also used with the corresponding

methods (push and pop)

* + 1. **Includes code written by students**

|  |
| --- |
| Code for exercise 2 |

* + 1. **Screenshots of executions and test carried out analyzing the results**

|  |
| --- |
|  |
| BFS successful in the tinyMaze |

|  |
| --- |
|  |
| BFS successful in the mediumMaze |

|  |
| --- |
|  |
| BFS successful in the bigMaze |

|  |
| --- |
| Since our search code is generic it works well on the 8-puzzle problem without changing it |

|  |
| --- |
| 7/7 in the q2 test provided |

* 1. **Conclusions on the behavior of pacman, it is optimal (y / n), reaches the solution (y / n), nodes that it expands, etc (0.5pt)**

The result is not optimal (there are better possible paths than the ones achieved).

The result reaches the solution.

The result expands all the nodes needed on a breath first algorithm, meaning it expands first the brothers.

* + 1. **Answer to question 3**

BFS does find the least cost solution. This does not mean it is the least computational cost.

## Section 3 (1 point)

* 1. **Personal comment on the approach and decisions of the proposed solution (0.5pt)**

To approach this question we worked around using a priority queue to implement the uniform cost algorithm in the easiest and strongest possible way.

* + 1. **List & explanation of the framework functions used**

To complete the uniform cost algorithm we used the provided framework functions. This consisted on:

* getStartState(): function to obtain the state the game begins in.
* isGoalState(state): function with a state as an argument. Returns boolean, true if the provided state is a goal state, else false.
* getSuccessors(state): function with a state as an argument. Returns a list with all the successors of the provided state.

The class PriorityQueue from the utils package was also used with the corresponding methods (push and pop)

* + 1. **Includes code written by students**

|  |
| --- |
| Code for exercise 3 |

* + 1. **Screenshots of executions and test carried out analyzing the results**

|  |
| --- |
|  |
| UniformCostSearch successful in mediumMaze |

|  |
| --- |
|  |
| StayEast with UniformCostSearch successful in mediumDottedMaze making the path choose the dots if possible |

|  |
| --- |
|  |
| StayWest with UniformCostSearch successful in mediumScaryMaze |

|  |
| --- |
| 7/7 in the q3 test provided |

* 1. **Conclusions on the behavior of pacman, it is optimal (y / n), reaches the solution (y / n), nodes that it expands, etc (0.5pt)**

The result is optimal in solution cost. It is not necessarily optimal in computational cost.

The result reaches the solution.

The result expands the nodes with the least cost function. If a node has already been visited, it will not be visited again.

## Section 4 (2 points)

* 1. **Personal comment on the approach and decisions of the proposed solution (1pt)**

To implement the A\* algorithm we used the uniform cost algorithm as a base and modified it to implement the heuristic use with values and comparisons.

* + 1. **List & explanation of the framework functions used**

To complete the A\* algorithm we used the provided framework functions. This consisted on:

getStartState(): function to obtain the state the game begins in.

isGoalState(state): function with a state as an argument. Returns boolean, true if the provided state is a goal state, else false.

getSuccessors(state): function with a state as an argument. Returns a list with all the successors of the provided state.

The class PriorityQueue from the utils package was also used with the corresponding methods (push and pop)

* + 1. **Includes code written by students**

|  |
| --- |
| Code for exercise 4 |

* + 1. **Screenshots of executions and test carried out analyzing the results**

|  |
| --- |
|  |
| bigMaze solved with A\* |

|  |
| --- |
| 7/7 in the q4 test provided |

* 1. **Conclusions on the behavior of pacman, it is optimal (y / n), reaches the solution (y / n), nodes that it expands, etc (1pt)**

The behaviour of pacman is not optimal. This is because elimination of repeated nodes can cause a better path to be evaded.

Pacman reaches the solution.

This algorithm expands the nodes with least cost and heuristic. This balance between the cost to the node and the path to the goal.

* + 1. **Answer to question 4**

|  |  |  |
| --- | --- | --- |
| A\* | BFS | DFS |

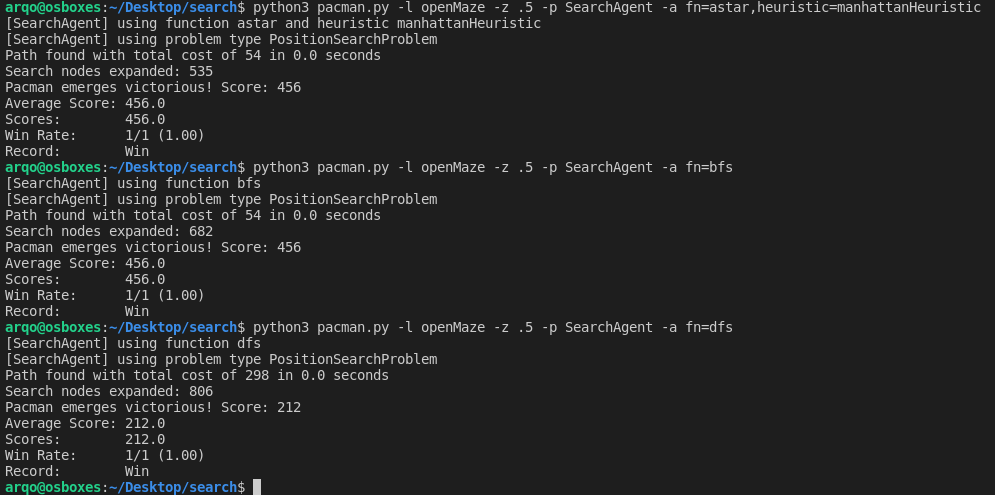
When we try to execute the openMaze with the three different algorithms developed, we obtain very different results: For the A\* algorithm with Manhattan heuristic gets a clean path shown by the colours (which represent the decision tree) where it is visible that it is an informed search because it follows a logical path to our eyes (for us who know the objective). On the other hand, for the BFS algorithm, almost the whole map is searched but the end path is the optimal one. Lastly, for the DFS , we see that neither the decision tree nor the path taken make sense to us.

The previous explanation can be interpreted also by the extended nodes count and the total cost of the solution shown as output which goes as following:

A\* : 535 nodes extended and a cost of 54

BFS : 682 nodes extended and a cost of 54

DFS : 806 nodes extended and a cost of 298

****

## Section 5 (2 points)

* 1. **Personal comment on the approach and decisions of the proposed solution (1pt)**

Question 5 made us formulate a new problem for the algorithms to solve. To approach this goal we created a list with the already accessed corners. This made it easy to add the corners when reached so we could know when the goal was achieved.

* + 1. **List & explanation of the framework functions used**

To complete this question we had to use various functions of the framework, including:

directionToVector(action): function to find the position reached after the corresponding action was performed.

* + 1. **Includes code written by students**

|  |
| --- |
| Code for exercise 5 |

* + 1. **Screenshots of executions and test carried out analyzing the results**

|  |
| --- |
|  |
| Successful solution to the corners problem in the tinyMaze |

|  |
| --- |
|  |
| Successful solution to the corners problem in the mediumMaze |

|  |
| --- |
| 3/3 in the q5 test provided |

* 1. **Conclusions on the behavior of pacman, it is optimal (y / n), reaches the solution (y / n), nodes that it expands, etc (1pt)**

The behavior of pacman is the expected one. It is optimal and it reaches the solution. The nodes it expands depend on the algorithm used.

## Section 6 (3 points)

* 1. **Personal comment on the approach and decisions of the proposed solution (1.5pt)**

To complete this question we made a heuristic with less cost the closest it was to the corners.

* + 1. **List & explanation of the framework functions used**

No interesting framework functions were used on this exercise.

* + 1. **Includes code written by students**

|  |
| --- |
| Code for exercise 6 |

* + 1. **Screenshots of executions and test carried out analyzing the results**

|  |
| --- |
| Successful feetback of test q4 provided |

|  |
| --- |
| Half the points for test q6 provided because we were unable to extend |

* 1. **Conclusions on the behavior of pacman, it is optimal (y / n), reaches the solution (y / n), nodes that it expands, etc (1.5pt)**
     1. **Answer to question 5: heuristics**

This heuristic tries to make pacman go to the corners. To achieve this we compare the distance from each tile to all the corners and use the biggest value as a reference to use. That’s how the heuristic will help the user to find the best possible path. The heuristic calculation only takes into account the corners left to discover.

## Section 7

This has been a very productive practice where we have been able to see in action all the knowledge about informed and uninformed search methods that we are acquiring in the theoretical classes. It has been very rewarding to see how the algorithms we provided were the basis for a game solving problem, on top of that, it was very easy to follow and provided extra information and hints that were very useful in order to obtain the functionality required. We have encountered some difficulties when it came to exercise 6 because we struggled to find an efficient heuristic that expanded less than 1200 and we found one (commented in the code) that should work but the autograder says it is inconsistent for some reason.

In a nutshell we are very fond of this practice and we reckon we have acquired all the necessary knowledge about the bases of the search algorithms.