16:332:573 - DATA STRUCTURES AND ALGORITHMS

COMPARISON BETWEEN PATH A* ALGORITHMS

...

NAME : PARTH KHARKAR
NET ID: pk674

TABLE OF CONTENTS

INTRODUCTION

- Path finding algorithms are an essential component of any modern system.
- These algorithms are basically used to help us find the shortest or the fastest between the source node and the destination node.
- In this presentation, we will be studying the implementation of A* algorithm, its variants Repetitive Forward A* algorithm and Adaptive A* algorithm.
- We will also be comparing Forward A* and Adaptive A* algorithm and study which one of these is more efficient and the key factors involved with them.
- The goal is to provide a comprehensive and unbiased comparison that can be used to solve the grid problem.

SEARCH ALGORITHM

- Search algorithms are computational techniques which are used to find an optimal path or search space.
- Search algorithms are commonly classified as uniformed or informed search algorithm depending upon whether they use domain knowledge or the heuristic search to guide the search process.
- It is a method to find solutions to problems by exploring various states of the problem domain.
- Search algorithm starts with an initial state in search of the solution of the problem, while searching for solutions it uses various data-structures such as breadth-first-search, depth-first search, or a heuristic search.
- There are various types of search algorithms and the A* algorithm is one of them.

A* ALGORITHM

- A* is a type of search algorithm also known as Heuristic search algorithms.
- A* algorithm is a popular informed search algorithm widely used for path-finding in a 2D or 3D environment.
- It is used to find the shortest distance or the optimal path between the source node and the destination node.
- A* algorithm works by expanding the nodes with lowest cost or the distance from the source node and a heuristic estimate of the distance to the destination node.
- The heuristic estimate is usually on Euclidean distance or the Manhattan distance between the node and the destination node.

ISSUE WITH A* ALGORITHM

- A* algorithm is not an adaptive algorithm (by which I mean to say - The agent needs to know the environment before it starts)
- What if the agent doesn't know about the environment and it needs to figure it out as it traverses.
- So, to inculcate this capability we use the Repetitive Forward A* algorithm and the Adaptive A* algorithm

REPETITIVE FORWARD A* ALGORITHM

- Repetitive Forward A* algorithm is an extension of the Forward A* algorithm and is particularly designed to handle the uncertainty in the environment.
- The idea behind implementing this algorithm is it runs the Forward A* multiple times from its current node to the destination node by always finding a new path when the previous path is found to be incorrect.
- If the path is found to be incorrect, a new A* algorithm search is started from the current node.
- The grid agent here does not know about the environment and figures it out while it is traversing the environment.

ADAPTIVE A* ALGORITHM

- Adaptive A* algorithm is another type of A* algorithm aimed to improve the efficiency as well as the accuracy of the algorithm by adjusting the heuristic function during every search.

- The idea behind using Adaptive A* algorithm is to use the information of its previous search and refine the heuristic estimate of the remaining cost from a node to the destination node.

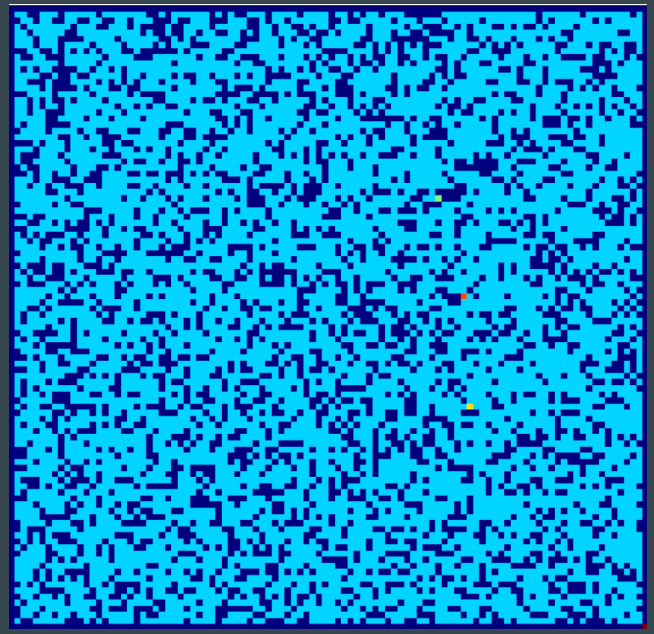
- This method helps us to improve accuracy of the heuristic function and can help us achieve a better performance.

- The heuristic function is updated after very search iteration based on the actual cost of the node to the destination node.

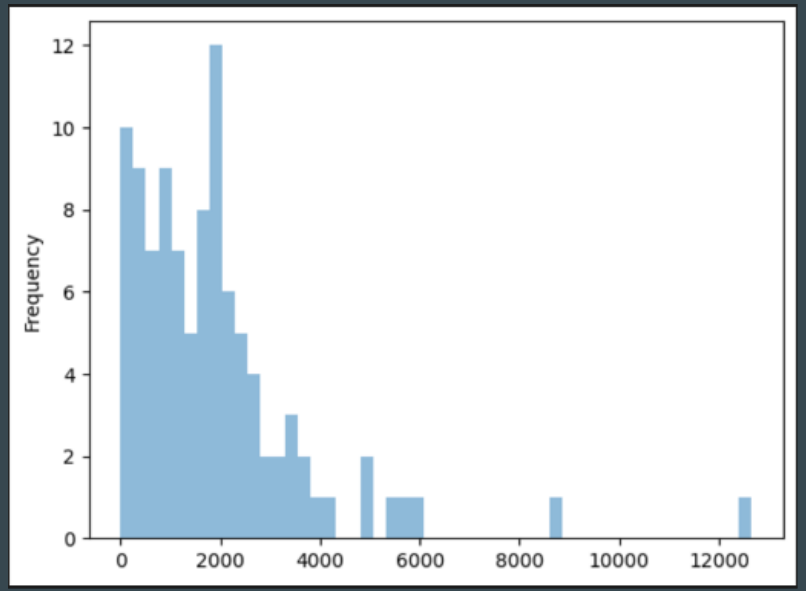
- The implementation is based on the previous search experiences.

EXPERIMENTAL SETUP

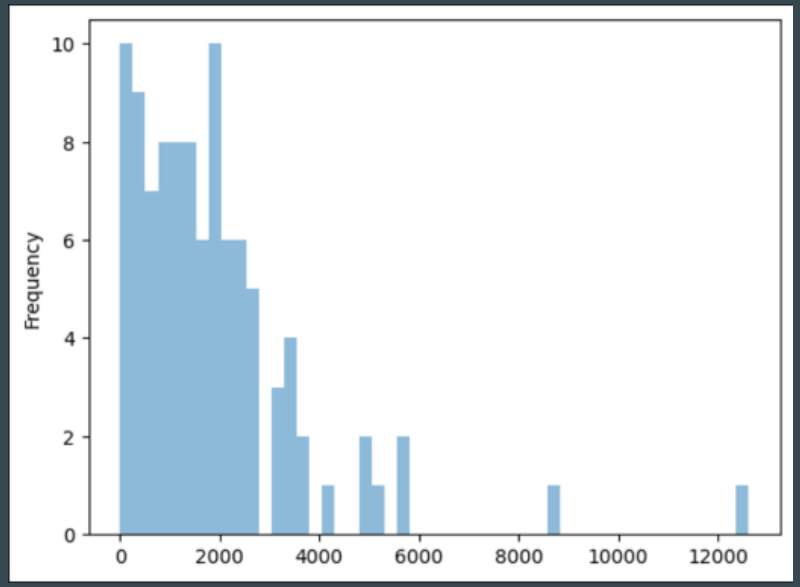
- To compare the Repetitive Forward A* and the Adaptive A* we would be creating 100 grid environments of size 100 x 100.
- We would be checking the total number of expanded cells in both the algorithms for all A* searches in a particular environment.
- The idea is to create a distribution of both algorithms and map out the total number of search based expanded cells.
- Using this information, we will run a “T-test” and generate a p value to prove our hypothesis of the difference between the two algorithms



GRAPHICAL REPRESENTATION



REPETITIVE FORWARD A* ALGORITHM



ADAPTIVE A* ALGORITHM

COMPARISON BETWEEN ALGORITHMS

```
T-statistic: 0.090147381955274 73
P-value: 0.46482865030341813
```

- Generating t-test provides a significant value, but the p-value does not provide an accurate representation based on our confidence interval of 0.05. Hence, we cannot prove that our means are significantly different.

```
THE NUMBER OF TIMES ADAPTIVE IS BETTER THAN FORWARD:  67
THE NUMBER OF TIMES FORWARD AND ADAPTIVE HAD A TIE:  22
THE NUMBER OF TIME FORWARD IS BETTER THAN ADAPTIVE:  11
```

- We use an assumption – the difference in individual expanded cell computations could theoretically tell which algorithm is more efficient. We prove through this method that Adaptive A* algorithm takes fewer computations, is therefore more efficient.

CONCLUSION

- In this project, I have experimented and compared to variants of the A* algorithm – Repetitive Forward A* Algorithm and the Adaptive A* Algorithm.
- Through the computations we tried to prove which one was more efficient than other.
- We tried two different methods to prove this, while implementing the first method we generated p-values to determine statistical differences between the algorithms. This method did not help us provide with a significant result.
- The other method which we employed was a simpler one, wherein we just calculated the difference between both the expanded cell computation.
- Through this methodology, we can come to significant result that Adaptive A* Algorithm is better in most of the instances