Search Algorithms Reflection

Charles Bailey

04/25/2015

CMPT 390

In Artificial Intelligence, one of our first major homework assignments was to implement 4 different search strategies for finding the quickest path through a maze. The four methods of searching through a maze were depth first, breadth first, uniform cost, and astar searches. This was one of the very first times where I programmed in python, as I had primarily written many programming assignments in Java or C language. I had some difficulty adjusting to a new language, as there were python specific issues I had to overcome. I searched online and asked Greg Gagne for help on many python issues. The most important challenge for me was differentiating the different search algorithms and figuring out how each one worked on the theoretical level. The breadth first algorithm worked by starting at a root node and then traversing the neighboring nodes in a level first before it moved to another level to search. The depth first algorithm revolved around starting at a root and then traversing down a branch before backtracking to another branch. While they are two different algorithms, I was surprised how in python there was only one character difference between switching from a queue to a stack. The depth first search had the line node = frontier.pop(), while the breadth first search algorithm had the line node = frontier.pop(0). If I had been programming in Java or C, I would have written different logic to implement the algorithms, and was surprised by how python simplifies using many data structures, and many less lines of code I had to write. This led me to write many programs in python whenever I could because of how simple python is and how one character changed the implementation of an algorithm.

The final two algorithms were a little harder to implement, as the uniform cost search needed to use a heuristic to find the shortest path between two nodes. The astar algorithm was similar to uniform cost search, but needed the cost of traversing a path in its calculations. I thought that I would have to write many lines of code to differentiate the two algorithms, but it turned out that one line of code was the difference between how to implement the algorithms. In the uniform cost search, I pushed the heuristic onto a priority queue, while the astar algorithm involved having the heuristic and cost pushed onto a priority queue. Once the algorithms were implemented correctly, we tested them by having each algorithm go through 6 mazes and seeing what the cost was to find the shortest path through. The results pointed to breadth first and astar strategies generally having the lowest costs, although the astar maze had astar and uniform cost search have the lowest costs. Writing the algorithms for this program gave me an opportunity to write an object oriented program and test four different algorithms to see which one had the most efficiency in finding the shortest path through a maze. While we had no unit tests, I learned that there are many ways of testing a program such as with text files or seeing how the costs of algorithms compare. The assignment also showed me that many programs can have methods achieve the same results, but that some algorithms are more efficient for certain tasks than others.