Andrew Melland

Prof. Clark

Introduction to Algorithms

10/9/2020

Searching Algorithms

For Lab 2 we coded and analyzed six different searching algorithms: Depth First Search (DFS), Breadth First Search (BFS), Dijkstra, A\*. We also made a iterative and recursive DFS and BFS search to analyze the impact that recursion and iteration has on performance time. DFS and BFS are also very similar algorithms with one bid difference. The only difference is that BFS uses a queue so that it searches all the closest elements to the starting node first, while DFS uses a stack so that it searches all the children first. Not only is BFS and DFS very similar but so is Dijkstra and A\*. While DFS and BFS just find a path to the destination, Dijkstra and A\* take into account the actual cost of traveling to the destination, finding the cheapest one. What separates Dijkstra and A\* is that A\* not only looks for the cheapest path but it also uses a heuristic cost that is estimated based on the next nodes distance to the destination, in this case, however the heuristic can be any value. This heuristic gives the A\* some basic intuition when it comes to deciding its next path which causes it to traverse in the correct direction, finding the destination node quicker than Dijkstra.

What we can see from our graph that the Dijkstra algorithm takes the longest times to execute, especially when the data set increases in size. This is expected because DFS and BFS do not consider the cost of the path and only care about finding the destination while Dijkstra a finds the cheapest path, which requires it to search every path that could possibly be the cheapest path. This means that as the data set increases, the number of paths that need to be calculated increases exponentially. Looking at the raw execution times for DFS and BFS it shows that as the data set size increases, the difference in execution time grows with BFS proving to be much quicker at finding the destination than DFS. DFS also explores more nodes than BFS and has a much larger path. This is because BFS is more likely to find a random node compared to if you knew the node was very deep, in which DFS would be much better.

We can also see that the time it takes to run the searches using an adjacency list is actually faster than the matrix, most likely because the adjacency list has only the children nodes connected to it while the adjacency matrix keeps track of all the children’s connections weather they are connected or not ie: 1 or 0. When finding all children nodes of a matrix, we have to loop through the number of nodes in the entire graph as opposed to only the children when using a adjacency list. A\* finds the initial path way way faster than Dijkstra because its heuristic points the algorithm in the right direction of the destination which cuts down on the execution time. The key to the heuristic is making its calculation time less than the time it actually saves by being calculated, it is entirely possible for a heuristic to negatively affect a search.

Sample Data

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adjacency List Source -> Destination | | | Search Algorithm | | | | | | | | | |
| DFS Iterative | | DFS Recursive | | DFS Iterative | | DFS Recursive | | Dijkstra | A\* |
|
| Avg Normalized Results | Nodes in Path | | 0.014 | | 0.014 | | 0.425 | | 0.425 | | 0.016 | 0.014 |
|
| Nodes Explored | | 0.33 | | 0.33 | | 0.407 | | 0.407 | | 0.523 | 0.011 |
|
| Execution Time | | 9E-04 | | 9E-04 | | 0.025 | | 0.025 | | 0.441 | 0.007 |
|
| Cost | | 0.015 | | 0.015 | | 0.427 | | 0.427 | | 0.008 | 0.011 |
|
|  |  | |  | |  | |  | |  | |  |  |
| Adjacency Matrix Source -> Destination | | | | Search Algorithm | | | | | | | | |
| DFS Iterative | | DFS Recursive | | DFS Iterative | | DFS Recursive | Dijkstra | A\* |
|
| Avg Normalized Results | | Nodes in Path | | 0.014 | | 0.014 | | 0.425 | | 0.425 | 0.016 | 0.014 |
|
| Nodes Explored | | 0.33 | | 0.33 | | 0.407 | | 0.407 | 0.523 | 0.011 |
|
| Execution Time | | 0.002 | | 0.001 | | 0.048 | | 0.048 | 0.436 | 0.008 |
|
| Cost | | 0.015 | | 0.015 | | 0.427 | | 0.427 | 0.008 | 0.011 |
|

Large 50

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adjacency List Source -> Destination | | Search Algorithm | | | | | |
| DFS Iterative | DFS Recursive | DFS Iterative | DFS Recursive | Dijkstra | A\* |
|
| Avg Normalized Results | Nodes in Path | 0.031 | 0.031 | 0.406 | 0.406 | 0.036 | 0.031 |
|
| Nodes Explored | 0.337 | 0.337 | 0.431 | 0.431 | 0.519 | 0.023 |
|
| Execution Time | 0.002 | 0.002 | 0.02 | 0.02 | 0.495 | 0.013 |
|
| Cost | 0.032 | 0.032 | 0.4 | 0.4 | 0.018 | 0.024 |
|

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adjacency Matrix Source -> Destination | | Search Algorithm | | | | | |
| DFS Iterative | DFS Recursive | DFS Iterative | DFS Recursive | Dijkstra | A\* |
|
| Avg Normalized Results | Nodes in Path | 0.031 | 0.031 | 0.406 | 0.406 | 0.036 | 0.031 |
|
| Nodes Explored | 0.337 | 0.337 | 0.431 | 0.431 | 0.519 | 0.023 |
|
| Execution Time | 0.001 | 0.001 | 0.026 | 0.026 | 0.484 | 0.01 |
|
| Cost | 0.032 | 0.032 | 0.4 | 0.4 | 0.018 | 0.024 |
|
|  |  |  |  |  |  |  |  |

Large 100

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adjacency List Source -> Destination | | | Search Algorithm | | | | | | | | | |
| DFS Iterative | | DFS Recursive | | DFS Iterative | | DFS Recursive | | Dijkstra | A\* |
|
| Avg Normalized Results | Nodes in Path | | 0.014 | | 0.014 | | 0.425 | | 0.425 | | 0.016 | 0.014 |
|
| Nodes Explored | | 0.33 | | 0.33 | | 0.407 | | 0.407 | | 0.523 | 0.011 |
|
| Execution Time | | 9E-04 | | 9E-04 | | 0.025 | | 0.025 | | 0.441 | 0.007 |
|
| Cost | | 0.015 | | 0.015 | | 0.427 | | 0.427 | | 0.008 | 0.011 |
|
|  |  | |  | |  | |  | |  | |  |  |
| Adjacency Matrix Source -> Destination | | | | Search Algorithm | | | | | | | | |
| DFS Iterative | | DFS Recursive | | DFS Iterative | | DFS Recursive | Dijkstra | A\* |
|
| Avg Normalized Results | | Nodes in Path | | 0.014 | | 0.014 | | 0.425 | | 0.425 | 0.016 | 0.014 |
|
| Nodes Explored | | 0.33 | | 0.33 | | 0.407 | | 0.407 | 0.523 | 0.011 |
|
| Execution Time | | 0.002 | | 0.001 | | 0.048 | | 0.048 | 0.436 | 0.008 |
|
| Cost | | 0.015 | | 0.015 | | 0.427 | | 0.427 | 0.008 | 0.011 |
|