Alex Wong

April 25, 2017

I chose to implement the algorithms in C# because I had heard that C# has a faster run time than Java and is also more efficient with memory. For each of the modified algorithm they relied heavily on a different list implementation. For the FIFO algorithm, I used a C# built in queue structure. For the stack, I also used C#’s built in stack data structure and for the dequeue, I chose to use the LinkedList implementation recommended by the textbook. I realize that the Min-Dist is supposed to be one on the fastest algorithms with List data structure being a priority queue based on the node weights. See the book says select the min distance, I went for the brute force way of iterating over the list to find the minimum know distance in the list of nodes. My implementation for most of these algorithms were to construct/populate an array of nodes that contain all the nodes in the graph. For simplicity I tried to make the arrays index be correspond to the same nodeID. A given Node would have list of arc objects; each arch object corresponds to a distance and an ending node. Since when I look at arcs I look at the arcs branch from a particular I would iterate over the list<Arc>. To generate the random ordering for examining Nodes for the Generic Algorithms. I generated a randomized list of integer where each integer corresponds to a Node in the Node array.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Arcs Scanned | Label Updates | Run-time |
| Generic Run 1 | 444,710,676 | 28,541,732 | 187.287 s |
| Generic Run 2 | 438,839,908 | 27,878,053 | 165.240 s |
| Average Generic | 441,775,292 | 28,209,892.5 | 176.2635 s |
| Modified FIFO | 77,239,918 | 29,950,702 | 16.72 s |
| Modified Stack | Can’t Finish | Can’t Finish  Very Slow | Could not finish |
| Modified Dequeue | 77,23918 | 29,950,702 | 21 s |
| Modified Min-Dist | 733,846 | 299,018 | 3.292 s |

The performance of each algorithm was measured based on how fast it could complete the shortest path for source node 1 on the NY Distance graph. Each array of Nodes was populated separately so no deep copying was necessary to generate new Node arrays. The data shown above represents my results after running each algorithm. Notice how the Modified Stack Algorithm could not complete. I allowed the algorithm to run for over 15 minutes when I came back the Stack had 3500 Nodes in it and it was on the 20 millionth iteration for trying to find the shortest path. The good thing is relative to my friend’s algorithm mine doesn’t fail because it is runs out of usable memory. The stack algorithm is just very slow. It would be expected that the Generic Algorithm would be the slowest because it is the closest thing to saying brute force the problem. My Fastest algorithms would be the Modified Min-Dist and my slowest algorithm would be the Stack because it could not even finish after running for 15 minutes. Modified Min Dist also examined and touched the least amount of Nodes and Arcs relative to the other algorithms.