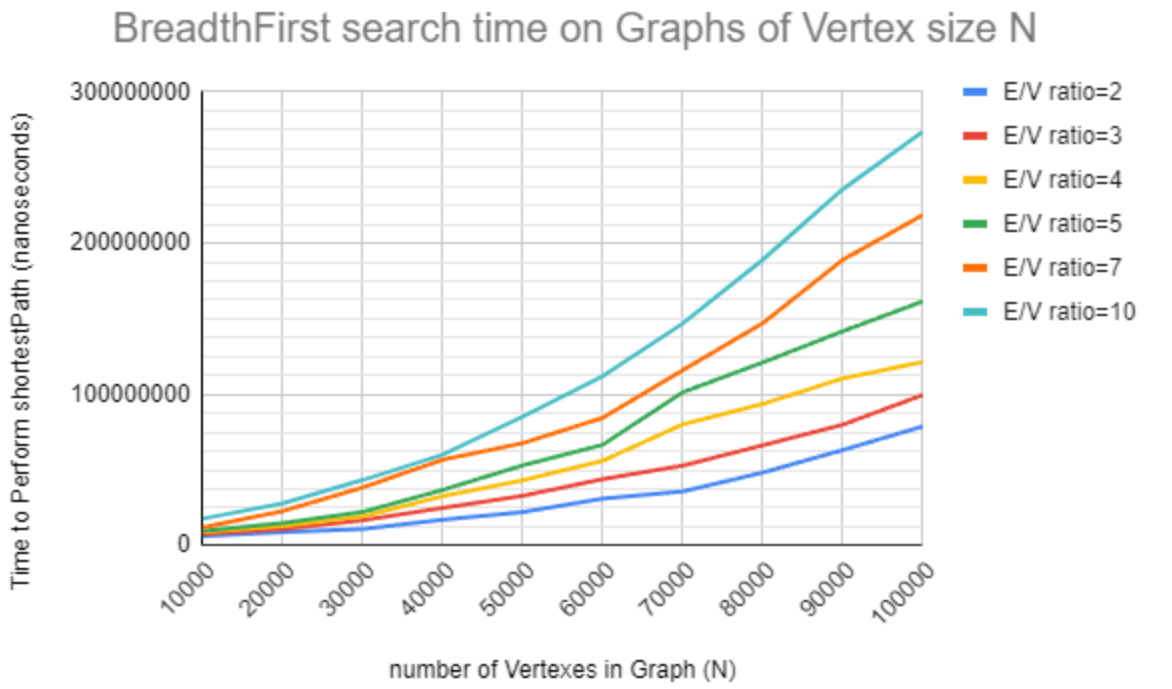
Hayden Walpole
U1210998
6/24/2024

Analysis Document — Assignment 7

1. I spent about 12 hours programming and testing my code for this assignment. Individual versus pair programming has a few differences, I think the code might have been faster to complete as a pair, but timing and debugging felt faster as an individual. I am not as confident in my code as an individual, but through testing I feel confident that is will work for most cases as intended. Although pair programming often offers more robust code, I think I prefer an individual programming approach better.

2. To assess the run-time efficiency of the Breath-first search method through the shortestPath method I designed an experiment that first initializes two empty arraylists and then feeds them into a method to create a random graph. The random graph method will build up a list of N number of vertices with values from a random seed input. It then has a for loop that adds directed edges between vertexes that are not themselves or duplicate edges for edgespervertex times by the total number of vertices. I used 10,000 vertexes as the first graph, up to 100,000 vertexes incrementing by 10,000 vertexes each iteration. We then call our shortestPath method from GraphUtlitiy and time the time it take to go from the vertex at index 0 at the source list, to a random vertex at index 0 to N in the source list. We used a times to loop of 500 for this experiment and found the average time to run the shortestpath method.



Because the speed of the shortestPath method depends on both the number of vertices and edges I analyzed the data for Big-O = O(|V|+|E|) where O(|E|) is expected to be between O(1) and O(V^2). From the following three analysis checks the overall behavior looks the most like O(|V|+|E|) when E is small, and as E gets larger the graph approaches O(|V|+|E|)^2. O(

	O(T) / O((V|+|E|)^2)					
V	E/V = 2	E/V = 3	E/V = 4	E/V = 5	E/V = 7	E/V = 10
10000	0.007052472222	0.008494096667	0.009941661111	0.01083612222	1.34E-02	1.98E-02
20000	0.002506547222	3.05E-03	3.60E-03	4.16E-03	6.34E-03	7.79E-03
30000	1.36E-03	2.08E-03	2.39E-03	2.75E-03	4.75E-03	5.35E-03
40000	1.20E-03	1.73E-03	2.27E-03	2.56E-03	3.95E-03	4.17E-03
50000	9.87E-04	1.47E-03	1.92E-03	2.36E-03	3.01E-03	3.79E-03
60000	9.61E-04	1.36E-03	1.73E-03	2.06E-03	2.61E-03	3.46E-03
70000	8.14E-04	1.20E-03	1.82E-03	2.30E-03	2.63E-03	3.33E-03
80000	8.39E-04	1.15E-03	1.63E-03	2.10E-03	2.55E-03	3.28E-03
90000	8.68E-04	1.10E-03	1.52E-03	1.95E-03	2.59E-03	3.23E-03
100000	8.77E-04	1.11E-03	1.35E-03	1.79E-03	2.43E-03	3.04E-03

	O(T) / O(|V|+|E|)					
V	E/V = 2	E/V = 3	E/V = 4	E/V = 5	E/V = 7	E/V = 10
10000	211.5741667	191.117175	178.9499	195.0502	2.41E+02	1.87E+04
20000	150.3928333	1.37E+02	1.30E+02	1.50E+02	2.28E+02	4.09E+04
30000	1.23E+02	1.40E+02	1.29E+02	1.49E+02	2.56E+02	6.18E+04
40000	1.43E+02	1.56E+02	1.64E+02	1.84E+02	2.84E+02	7.70E+04
50000	1.48E+02	1.65E+02	1.73E+02	2.12E+02	2.71E+02	1.03E+05
60000	1.73E+02	1.83E+02	1.87E+02	2.22E+02	2.82E+02	1.22E+05
70000	1.71E+02	1.89E+02	2.29E+02	2.90E+02	3.31E+02	1.55E+05
80000	2.01E+02	2.08E+02	2.34E+02	3.03E+02	3.68E+02	1.82E+05
90000	2.34E+02	2.22E+02	2.46E+02	3.15E+02	4.20E+02	2.12E+05
100000	2.63E+02	2.49E+02	2.43E+02	3.23E+02	4.37E+02	2.20E+05

	O(T) / O(LOG(|V|+|E|))					
V	E/V = 2	E/V = 3	E/V = 4	E/V = 5	E/V = 7	E/V = 10
10000	0.001575224753	0.000000283136	0.000000331388	0.000000361204	4.47E-07	-3.27E-03
20000	0.000524585156	5.08E-08	6.00E-08	6.93E-08	1.06E-07	-1.14E-03
30000	2.75E-04	2.31E-08	2.66E-08	3.06E-08	5.28E-08	-7.48E-04
40000	2.35E-04	1.45E-08	1.90E-08	2.13E-08	3.29E-08	-5.67E-04
50000	1.91E-04	9.78E-09	1.28E-08	1.57E-08	2.01E-08	-5.02E-04
60000	1.83E-04	7.55E-09	9.63E-09	1.14E-08	1.45E-08	-4.52E-04
70000	1.53E-04	5.72E-09	8.66E-09	1.10E-08	1.25E-08	-4.29E-04
80000	1.56E-04	4.80E-09	6.78E-09	8.76E-09	1.06E-08	-4.19E-04
90000	1.60E-04	4.07E-09	5.62E-09	7.20E-09	9.60E-09	-4.09E-04
100000	1.60E-04	3.69E-09	4.50E-09	5.98E-09	8.10E-09	-3.82E-04

3. A Breadth search algorithm is expected to have a run-time efficiency between O(1) and O(|V|^2) based on how many edges the graph contains. In lecture we stated that this is often O(|V|LOG|V|+|E|). Our implementation performs close to as expected, but alittle slower, as we

never saw O(1) behavior as we increased edges, and the time took longer with more edges. This is what we would expect since we are using an adjacency list which will work better for sparse graphs. This implementation also could be slower due to our choice of using a HashMap, or Arraylists in a place where a different data structure could be faster. The relationship between |V| and |E| will affect the run time by making it longer as |E| increase for the same size |V|.

4. If designing an experiment for timing a depth-first Search we need to be careful about stack overflow errors with recursion. These will likely occur when we have long chains of Vertices in a path. This would occur if our graphs were very large and had few edges, acting almost as a list. If the algorithm handled cycles, but the cycles were long we could also hit a stack overflow from recursion before the cycle was discovered. A graph with a lot of edges per vertex can also cause stack overflow if our depth-first search is called on lots of branches recursively. The best approach would be to find a lower and upper limit that your edges per vertex could stay between, while also limiting our graph size.

5. When designing an experiment for timing a topological sort we need to consider what our desired results are. Often a topological sort is used with a graph that is directed, and cycle free. We would also want the graph to have a natural hierarchy so that there were natural groups of vertices that were at the beginning, middle, and end of our data groups like the sandwich making instruction example in class. If the vertices are all randomly connected we will still get a topological ordering, but the use of this sort would not be very practical. By limiting our graphs in experiment from being completely random, to these cases we could best understand the other characteristics of the graphs that effect the timing.