Austin Johnson

User: agjohns

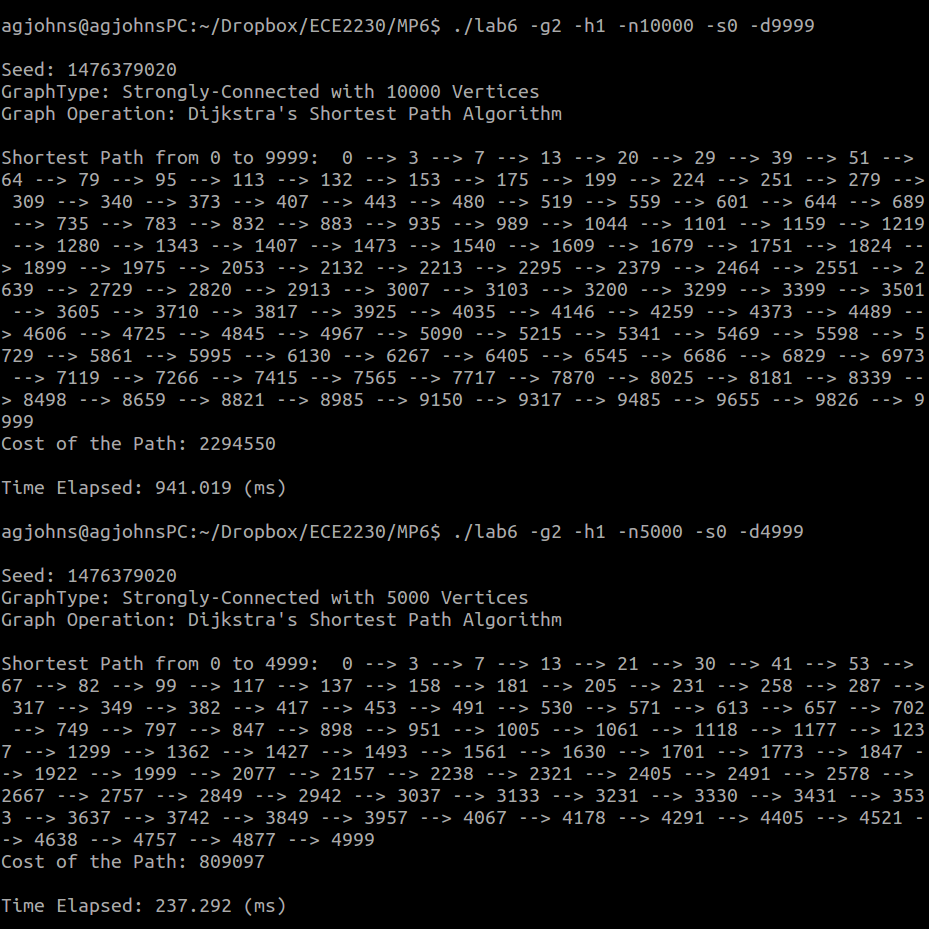
Machine Problem 6

16 November 2016

Performance Evaluation

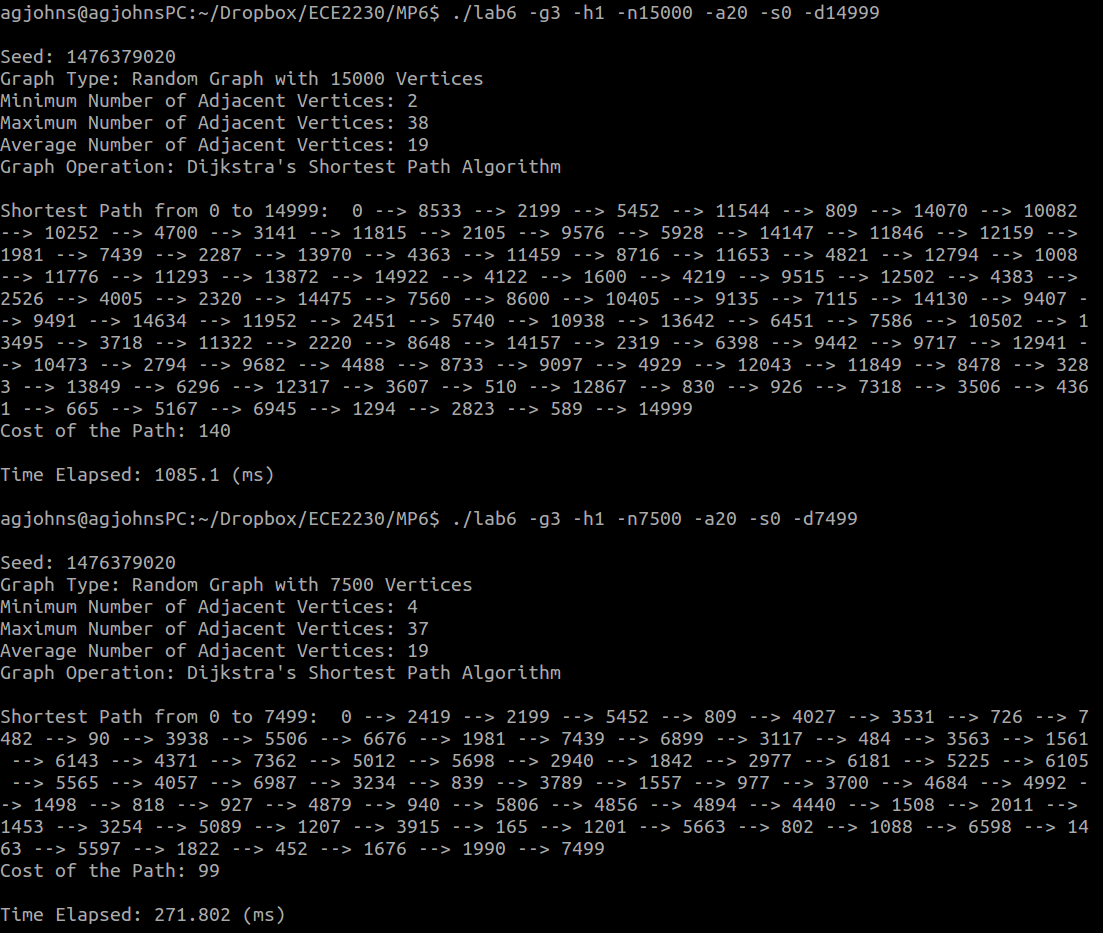
**Implementation:** I chose to implement a graph using an adjacency list over an adjacency matrix. The reason I did this was because most real world applications use sparse graphs to represent data and an adjacency list would be a more practical implementation due to conserving memory. It was not very hard to do and I did see the benefits of using a list over a matrix in almost all cases except when I was using Graph 2 which is strongly connected. I noticed that when I ran valgrind and compare with the adjacency matrix version I dynamically allocated a lot more memory (almost double) than the other implementation.

1.a Show that your implementation for Dijkstra’s Algorithm is O(n^2) for Graph 2:



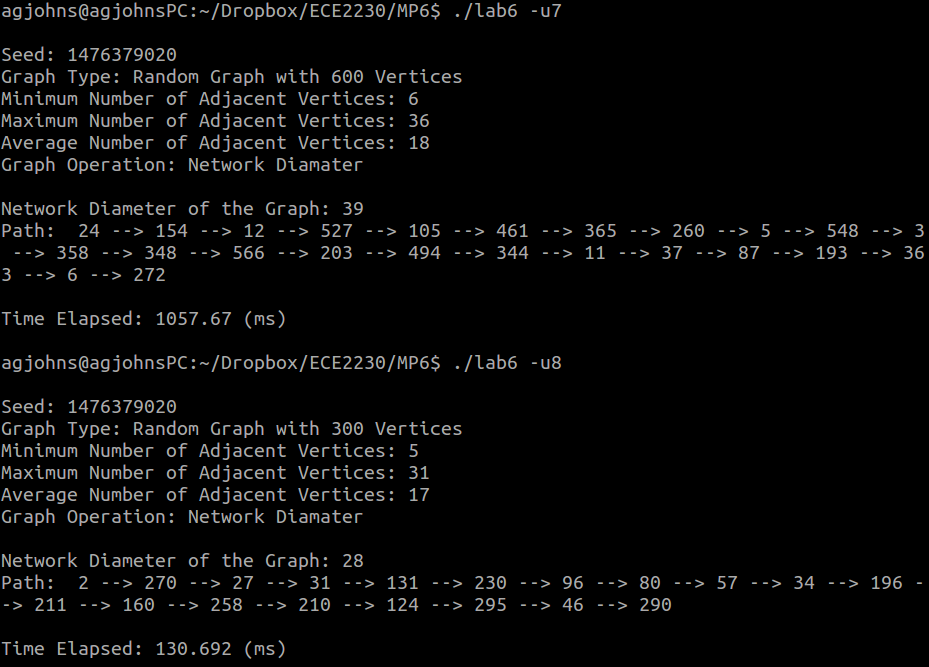
The ratio of my two experiments is: which is approximately 4. The ratio of the two numbers being approximately 4 verifies that my implementation is O(n^2) because we are dividing by 2  (2)^2 equals 4.

1.b Show that your implementation for Dijkstra’s Algorithm is O(n^2) for Graph 2:



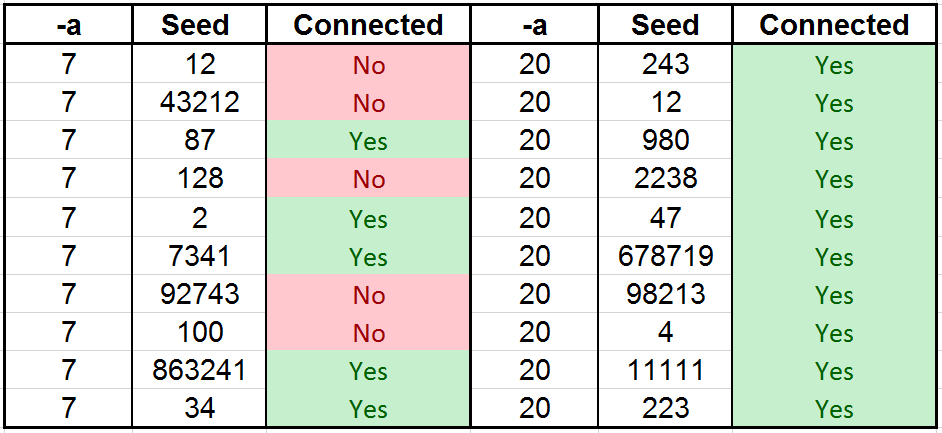
The ratio of my two experiments is: which is approximately 4. The ratio of the two numbers being approximately 4 verifies that my implementation is O(n^2) because we are dividing by 2  (2)^2 equals 4.

2. Verify that your Network Diameter implementation is O(n^3) for Graph 3 using –a20:



The ratio of my two experiments is: which is approximately 8. The ratio of the two numbers being approximately 8 verifies that my implementation is O(n^3) because we are dividing by 2  (2)^3 equals 8. My implementation is O(n^3) because I am using a for loop to set each vertex in the graph as the source and then calling my Dijkstra’s algorithm, which is O(n^2). This ends up being O(n) \* O(n^2) which equals O(n^3).

3. Verify that a higher number of adjacent vertices in Graph 3 equals a higher probability of being a connected graph:



4.a Show that the number of paths for Graph 2 is equal to N-1 where N is equal to the number of vertices:

Seed: 1476379020

Graph Type: Strongly-Connected with 15 Vertices

Graph Operation: Multiple Link-Disjoint Paths

Multiple Link-Disjoint Paths From 14 to 0:

Link-Disjoint Path 1: 14 --> 8 --> 3 --> 0

Cost of the Path: 185

Link-Disjoint Path 2: 14 --> 9 --> 4 --> 0

Cost of the Path: 187

Link-Disjoint Path 3: 14 --> 7 --> 2 --> 0

Cost of the Path: 187

Link-Disjoint Path 4: 14 --> 10 --> 5 --> 1 --> 0

Cost of the Path: 192

Link-Disjoint Path 5: 14 --> 6 --> 0

Cost of the Path: 196

Link-Disjoint Path 6: 14 --> 5 --> 0

Cost of the Path: 199

Link-Disjoint Path 7: 14 --> 11 --> 7 --> 0

Cost of the Path: 210

Link-Disjoint Path 8: 14 --> 12 --> 8 --> 0

Cost of the Path: 226

Link-Disjoint Path 9: 14 --> 13 --> 9 --> 0

Cost of the Path: 246

Link-Disjoint Path 10: 14 --> 0

Cost of the Path: 270

Link-Disjoint Path 11: 14 --> 4 --> 10 --> 0

Cost of the Path: 372

Link-Disjoint Path 12: 14 --> 3 --> 6 --> 11 --> 0

Cost of the Path: 438

Link-Disjoint Path 13: 14 --> 2 --> 6 --> 12 --> 0

Cost of the Path: 506

Link-Disjoint Path 14: 14 --> 1 --> 3 --> 7 --> 13 --> 0

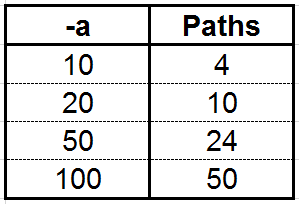
Cost of the Path: 580

No other path exists.

Time Elapsed: 0.103 (ms)

There are 15 vertices in the graph. The number of paths were 14 → Paths = N - 1

4.b Compare the number of link-disjoint paths found using varying values of adjacent neighbors (-a R) for Graph 3:



Notice that as the number of adjacent neighbors increases, the number of link-disjoint paths increases. The number of paths is approximately equal to the number of adjacent neighbors divided by 2 