Hrithik Shivnauth
DS210
Final Project Writeup – Six Degrees of Separation

In my project, I used a Facebook dataset to analyze the usual distances between friends on the website.

The essence of my project is describing a unique way that the distances between the vertices are separated and what degree (number of other nodes connected to a current node) would determine different variants of separation between nodes. My program takes an input of 4039 nodes connected using 88,234 edges. This project is meant to understand how graphs can be used to show relationships between others when doing a deep analysis of distance metrics.

My main function calls 4 different functions

- Different metrics analyze the way the edges and vertices are interconnected.

These 4 functions are :

BFS from every Node : Calculate for each individual node the number of nodes it is connected to and the distance edges outgoing from each connection.

An example output.

Distances from node 3249 -> {1097: 3, 316: 4, 1483: 3, 3630: 5, 2184: 4, 3679: 5, 2232: 4, 3971: 5, 1170: 3, 717: 5, 3239: 2, 1592: 3, 477: 4, 1335: 3, 2211: 4, 2348: 4, 623: 4, 3877: 4, 42: 4, 1196: 3, 1422: 3, 114: 4, 970: 3, 1384: 3, 2912: 2, 2182: 4, 2879: 2, 3051: 2, 3487: 5, 289: 4, 1455: 3, 1156: 3, 513: 4, 3960: 5, 1872: 3, 844: 5, 3297: 2, 906: 3, 2015: 4, 451: 4, 2892: 2, 2614: 4, 3377: 2, 380: 4, 2596: 4, 1252: 3, 3584: 5, 2550: 4, 558: 4, 3788: 5, 2386: 4, 3932: 5, 357: 4, 621: 4, 1830: 3, 1175: 3, 1793: 3, 2954: 2, 2516: 4, 1909: 3, 2790: 2, 2284: 4, 467: 4, 3789: 5, 3643: 5, 55: 4, 450: 4, 2758: 2, 1278: 3, 121: 4, 625: 4, 2823: 2, 3069: 2, 3365: 2, 3439: 5, 2254: 4, 1362: 3, 2539: 4, 3285: 2, 4033: 6, 1760: 3, 3145: 2, 1168: 3, 2024: 4, 335: 4, 2624: 4, 2213: 4, 2512: 4, 1054: 3, 1032: 3, 723: 5, 1118: 3, 1604: 3, 226: 4, 3568: 5, 3375: 2, 3071: 2, 3927: 5, 1230: 3, 828: 4, 2291: 4, 1212: 3, 949: 3}

In this output, you can notice that node 3249 is connected to node 1037 by a distance of 3 edges, and 316 by 4 edges, and 1483 by 3 edges and so for forth.

Distance Statistics for every Node: This function calculates the total number of nodes at each distance 1 through 8 for every node in the graph. It gives us an in-depth insight into the local structure and connectivity from each node's perspective.
This is a current sample output:

```
Node 1418: Distance counts up to 8:
  Distance 1 -> 21
  Distance 2 -> 1024
  Distance 3 -> 1641
  Distance 4 -> 1093
  Distance 5 -> 117
  Distance 6 -> 142
  Distance 7 -> 0
  Distance 8 -> 0
Node 1465: Distance counts up to 8:
  Distance 1 -> 108
  Distance 2 -> 1690
  Distance 3 -> 1462
  Distance 4 -> 519
  Distance 5 -> 117
  Distance 6 -> 142
  Distance 7 -> 0
  Distance 8 -> 0
Node 904: Distance counts up to 8:
  Distance 1 -> 3
  Distance 2 -> 1053
  Distance 3 -> 1630
  Distance 4 -> 1093
  Distance 5 -> 117
  Distance 6 -> 142
  Distance 7 -> 0
  Distance 8 -> 0
Node 1263: Distance counts up to 8:
  Distance 1 -> 6
  Distance 2 -> 1039
  Distance 3 -> 1641
  Distance 4 -> 1093
  Distance 5 -> 117
  Distance 6 -> 142
  Distance 7 -> 0
  Distance 8 -> 0
Node 1547: Distance counts up to 8:
  Distance 1 -> 122
  Distance 2 -> 923
  Distance 3 -> 1641
  Distance 4 -> 1093
  Distance 5 -> 117
  Distance 6 -> 142
  Distance 7 -> 0
  Distance 8 -> 0
Node 2790: Distance counts up to 8:
  Distance 1 -> 9
  Distance 2 -> 783
  Distance 3 -> 1038
  Distance 4 -> 1496
  Distance 5 -> 657
  Distance 6 -> 55
  Distance 7 -> 0
  Distance 8 -> 0
```

One can notice, that there is a total of 8 different distances that are computed after conducting a breath-first search analysis. From that point at each distance, for example, Node 1418 has 21 nodes at distance 1 that are outgoing from the graph, while having 1024

nodes at distance 2. We can now conduct further analysis to see which distance is considered the most important.

With these results, we can ask the question of which distance holds the most priority by seeing maximum number of distances for each one of the eight distances.

Max_distance_Counts: This function is useful for computing the maximum distance counted for every single possible distance entry of 1 through 8. It applies a for loop, that iterates through every single node in the graph and counts the occurrences of distances 1 through 8 for every node and all the nodes it is connected to. It updates a variable called total counts, which calculates the total number of distances for every available distance 1-8 across all nodes in the graph.

Finally it calculates the frequency by:

- first iterating through the breath first search print statement to find the total number of nodes with a particular distance.
- then it finds the maximum distance that that individual node is connected to.
- Next, it divides the nodes with a certain maximum distance either 1 through 8 by the total count for each distance 1 through 8.

In this case, the frequency represents a percentage that is meant to show the rate of appearance of that distance among connected nodes within the graph. Once looking at these rate we can see which distance holds the most strength in holding the graph together and further analysis could understand why those distances are more common.

We can also, get a good picture to understand the way nodes are separated, for example, if the frequency for distance 4 is lower than the frequency for distance 5, we can show frequency 4 has more connectedness and less separation than frequency 5.

The distance counts function is a good comparison as we can see the number of nodes connected at a certain distance at a particular node. compared to the total number of times that distance appears in the graph.

If we compare it to the frequency, we can see the total number of nodes connected to a certain distance at a particular node when compared to the weight of the level of separation that the distance holds throughout the graph. We can also use this information to understand why certain nodes have more nodes connected to them at distance 1 than another node at distance 1 if distance 1 holds a low separation frequency showing high connectedness in the graph.

```
Distance 1: Max count = 176468, Frequency = 0.0059217535190516125
Distance 2: Max count = 2716134, Frequency = 0.0010687985202497373
Distance 3: Max count = 3981852, Frequency = 0.0006604966734072487
Distance 4: Max count = 5861560, Frequency = 0.0004190010850353831
Distance 5: Max count = 2565170, Frequency = 0.0007282168433281225
Distance 6: Max count = 677214, Frequency = 0.0024408827933267772
Distance 7: Max count = 315464, Frequency = 0.0034425481195952626
Distance 8: Max count = 15620, Frequency = 0.00909090909090909
```

For example, here we can see that distance 4 holds a very high amount of connectedness since it has a low separation frequency, since the number of times that a node has 4 as a maximum distance divided by the total number of nodes with a distance of 4 is low.

```
Node 2851: Distance counts up to 8:
  Distance 1 -> 22
  Distance 2 -> 770
  Distance 3 -> 1038
  Distance 4 -> 1496
  Distance 5 -> 657
  Distance 6 -> 55
  Distance 7 -> 0
  Distance 8 -> 0
```

Average Number of Nodes at each distance – this computes the average number of nodes that appear at a particular distance.

Average distance- this computes the average distance after iterating between a sample of 3,000 nodes throughout the graph.

```
Number of vertices: 4039
Average distance: 3.714225055706858
Average number of nodes at each distance:
Distance 1: 43.69
Distance 2: 672.48
Distance 3: 985.85
Distance 4: 1451.24
Distance 5: 635.26
Distance 6: 172.49
Distance 7: 234.20
Distance 8: 79.29
```

Finally, in my main function, I run a series of tests for the function average distance and test BFS for distances between single nodes and chained distances.