Assignment 5

Junzhu Xiang, Haomin Lin

Part 1:

In part 1, a random scale-free network graph is generated. This network graph contains the number of N nodes, and it has few hub nodes connecting to bunch of other nodes. And each connection has a weight. I will show the correctness of part 1 below.

(1) The correctness of input. If number of nodes are less than or equal to 0(N<=0), the program will raise error. And the input weight should be larger than 0. W2 must be larger than or equal to W1. In the command line, the second parameter is N. Third and fourth parameter is W1 and W2. Testing results show below. Also, in this program, if W2 equals to W1, it can still generate weight, but all weights are the same.

```
-bash-4.1$ ./graphgen -100 1 2 -o top.txt -h histo.txt
This program is: ./graphgen
Error! N must be a positive integer!
```

```
-bash-4.1$ ./graphgen 100 0 2 -o top.txt -h histo.txt
This program is: ./graphgen
Error! w_1 should be larger than 0!
```

```
-bash-4.1$ ./graphgen 100 2 1 -o top.txt -h histo.txt
This program is: ./graphgen
Error! w_2 must be larger or equal to w_1!
```

(2) Generate the graph. Let's set N = 10. The picture below shows the graph, which is stored and expressed by linked lists.

```
C:\Users\jzhsi\Desktop\CSE6010\A5>graphgen 10 0 5 -o top.txt -h histo.txt
This program is: graphgen

The graph is below:
0 1 2
1 2 0 3
2 0 1 4 5 6 7 8 9
3 1
4 2
5 2
6 2
7 2
8 2
9 2
```

As you can see, in the terminal, the first data in each line is the data of each vertex. After the first data, there are some other nodes connecting to this vertex. It clearly shows that the vertex[2] can be defined as a hub of this free-scale network graph, because the number of its edge nodes is the most. If you let N become larger, like N = 1000, you can see there are more than one hub in this graph.

Meanwhile, according to the instruction, at first, the graph contains 3 nodes, and the graph is fully connected. So the first three nodes are connected with each other. As you can see the picture above, nodes 0, 1, 2 have fully connected.

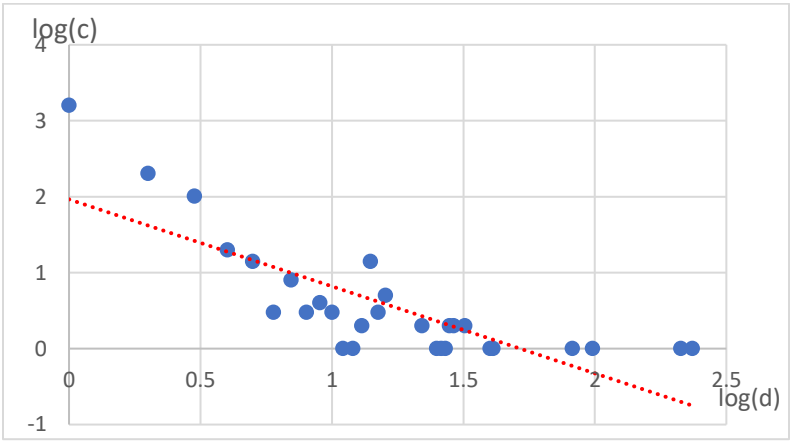
Also, we can output a histogram file to verify the correctness of this program. Let's set N = 2000, after running this program, we can get the histo.txt file. In this file, the data format is the same as what the instruction said. Below is part of this output file.

histo.txt - Notepad

File Edit Format View Help

```
2 202 0 3 22 27 29 30 46 47 49 51 53 66 67 70 73 74 81 82 88 89 93 320 321 322 325 326 327 328 329 330 331 337 340 341 342 344 345 346 347 350 351 352 354 35
15 3 1 71 86
212 1 2
1 1598 4 6 7 8 9 10 11 12 13 14 16 18 19 20 21 32 33 34 35 36 38 39 41 42 43 44 45 55 57 58 59 60 61 62 63 64 65 75 76 77 78 79 80 90 91 92 114 115 116 117 1
392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 4
692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 742 743 744 745 7
1 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060
97 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334
537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 157
1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 18
1995 1996 1997 1998 1999
14 14 5 17 37 40 127 134 224 237 324 332 349 353 368 735
40 1 15
28 2 23 367
16 5 24 72 85 87 95
235 1 25
3 102 26 28 31 48 50 52 54 68 69 83 84 94 96 97 98 99 100 102 113 343 348 356 360 373 460 536 610 611 613 644 647 649 650 681 682 683 684 687 710 711 712 714
26 1 56
4 20 101 107 110 323 361 538 570 571 572 573 612 614 648 651 680 783 791 815 817 876
41 1 103
9 4 104 339 532 788
29 2 105 786
22 2 106 370
32 2 108 111
5 14 109 369 458 500 501 533 537 539 583 642 734 784 789 790
98 1 112
25 1 182
13 2 265 463
27 1 299
8 3 338 412 643
7 8 357 362 366 372 411 459 685 686
10 3 358 584 787
11 1 363
82 1 371
6 3 499 769 785
```

In each line, the first data the node of degree i, the second is the number of nodes in the network with degree i, and after that it is a list of nodes. With Microsoft Excel, we can use these data to verify this network is scale-free. After adding trendline, we can see these data's distribution follow the power law.



(3) Data format in the top.txt. To complete part 2, an output file *top.txt* should be passed to part 2 program. An overlook of this file is below: (Let's set N = 10)

top.txt - Notepad

File Edit Format View Help

```
10
0 2 1 0.531785 2 1.937620
1 3 2 1.937620 0 0.531785 3 2.286447
2 8 0 1.937620 1 1.937620 4 1.747185 5 1.724448 6 4.572893 7 1.609699 8 4.483779 9 4.218116
3 1 1 2.286447
4 1 2 1.747185
5 1 2 1.724448
6 1 2 4.572893
7 1 2 1.609699
8 1 2 4.483779
9 1 2 4.218116
```

This first line contains one data, which is the number of nodes(N) in this graph. From the second line, the first data is the index of vertex[i], and the second data is the degree of vertex[i]. After that, every two data

are grouped together. For example, the third data in this line is the index of one edge node of vertex[i], and the fourth data is the weight between the edge node and vertex[i] and so forth.

Part 2: Application analysis

One important application of the scale free network is airport network. Airport is very important for the transportation system of a country. So many countries plan its transportation network by analyzing the scale free network. For example, the Italian airport network is a scale free network. To prove that, researchers have collected data from OAG Max database including all the scheduled flights and scheduled charter flights of the world's airlines for the period June 1, 2005, to May 31, 2006 within Italy, and they calculate for each main city's airport the connection degree, that is the number of other airports to this airport is connected by a non-stop flight[1]. After that, they get a graph of the Italian airport network.

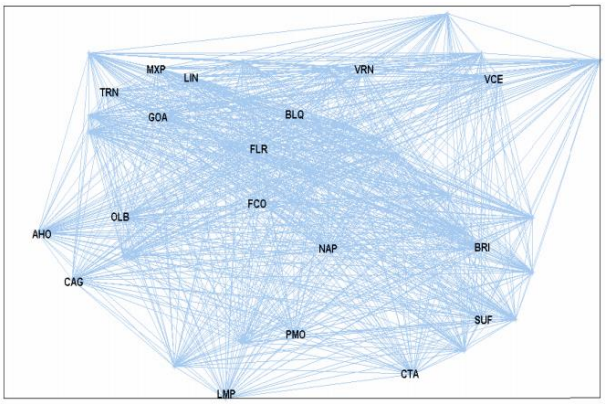


Fig.1: The Italian Airport Network (IAN)

Source: Quartieri J, Guida M, Guarnaccia C, et al. Complex Network Applications to the Infrastructure Systems: the Italian Airport Network case[J]. New Aspects of Urban Planning and Transportation, 2008.

As you can see, in the figure above, airports with large quantities of airlines in Italian main cities can be considered as hubs of the scale free network. There are also some leaf nodes in the Italian airport network. These leaf nodes are normally smaller airports with fewer airlines. In this scale free Italian airport network, every link among these nodes can represent every airline, and the weight on the link can be considered as the distance from one airport to the other.

Also, this graph can answer some questions. If a city like Rome is the hub node, we can know that Rome must be an important city in Italy, and the number of tourists in this city can be very large. So we will have an overview of its development in economy, tourism, etc. Meanwhile, this graph can help plan the location of a new airport in this country. If one hub node has too many airlines, which are much larger than any other hub nodes' airlines, people can decide to build a new airport near that hub node, so it can reduce the traffic pressure of that airport. Also, with this graph, airline companies can properly design their airlines and arrange their fights.

The preferential attachment rule is appropriate for this airport network. The reason is that in the network, rich nodes must be those large cities, because they can attract quantities of people to visit, and many factories are in main cities, so they have to transport many cargoes to other places. Therefore, main cities should have the larger airport with more airlines than other smaller cities, and if the city becomes larger, its airport should have more airlines.

Part 3: Graph analysis

By implementing Dijkstra algorithm, we can analyze the graph by calculating its diameter and the longest distance from any source node to any other nodes in the graph and printing the path of the longest distance from the source node. The output file name can be defined by entering the execution command as "% analysis topology –o output", in which "analysis" is the execution file name, "topology" is the source file name of a graph, and "output" stands for the output file name.

```
lawn-143-215-54-199:data humas$ ./analysis 0R.txt -o out.txt
This program is: ./analysis
Output file is: out.txt
```

In the program, the program made for Assignment two is reused with modification. And the process of analyzing a graph is shown below:

(1) Importing the data. The data from Part 1 can be configured like this:

```
5
0 2 1 420 2 175
1 2 0 420 3 150
2 2 0 175 3 400
3 3 1 150 2 400 4 100
4 1 3 100
```

The first line shows the number of nodes in the graph. After that, each line shows the node information, which includes source node number, degree and the information of its links. So, for every line, the first two number stands for source node number and degree respectively. The program implements an array storing n priority queues, n is the degree of the source node. The priority queue stores the information of all the nodes linking to the source node. And each node in the priority queue stores the information of each node linking to the source node. It can be shown like this:

```
Source: 0
2 : 175.000000, parent: 0
1 : 420.000000, parent: 0

Source: 1
3 : 150.000000, parent: 1
0 : 420.000000, parent: 1

Source: 2
0 : 175.000000, parent: 2
3 : 400.000000, parent: 2

Source: 3
4 : 100.000000, parent: 3
1 : 150.000000, parent: 3
2 : 400.000000, parent: 3

Source: 4
3 : 100.000000, parent: 4
```

(2) Calculating the distance.

After assigning all the data, the next step is to calculate the distance between the source node and any other nodes in the graph. Firstly, we relax all the nodes but the source node, setting the original distance as infinity.

```
starting 0
Distance for 0: 0.000000
Distance for 1: inf
Distance for 2: inf
Distance for 3: inf
Distance for 4: inf
```

Then we'll start from the source node to calculate the distances to its neighbors, and then store the distance associated with the neighbor's number and the neighbor's front node in another priority queue, so that we can put the nearest node in the front. Then we will turn to the nearest neighbor to calculate the distances to its neighbors and do the same thing and mark it as visited after the calculating is done, so that the program will skip the visited node when encounter it again.

```
2 : 175.000000, parent: 0
1 : 420.000000, parent: 0

Node 2 visited
Source: 0
1 : 420.000000, parent: 0
3 : 575.000000, parent: 2
```

With all nodes in the graph except the source marked, the calculation will end and the priority queue storing useless distances will be freed. Since all the nodes will be freed in the end, the program implements an array to store the final result of distances:

```
0's distance: 0.000000
1's distance: 420.000000
2's distance: 175.000000
3's distance: 570.000000
4's distance: 670.000000
```

(3) Drawing the longest path

When all the other nodes have been calculated (visited), we will locate the node with largest distance from the source node and then use its front node information to draw the path from the source to it. We'll also store the nodes on the path in a priority queue, so that the nearest node can be printed out first. And the program will also associate the longest distance with the source node in the structure.

```
Path for 0:
Farthest Node #4:
Its Longest Distance:670.000000
Path: 0->1->3->4
```

(4) Write the output file

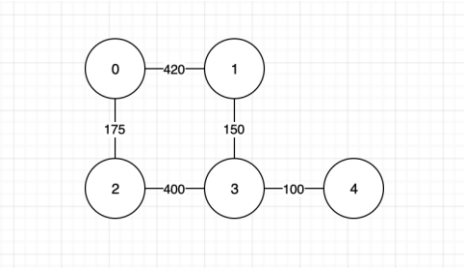
To write the diameter of the graph, we will compare all the longest distances of each node and the longest one. Then, starting from 0, the program gets the node's longest distance from the priority queue storing its longest path, and then print out the nodes on the path to the output file. And the output file is like the following:

out.txt

```
The diameter is: 670.000000
Node #0:
The Longest distance of it:670.000000
The Longest path of it:0->1->3->4
Node #1:
The Longest distance of it:550.000000
The Longest path of it:1->3->2
Node #2:
The Longest distance of it:550.000000
The Longest path of it:2->3->1
Node #3:
The Longest distance of it:570.000000
The Longest path of it:3->1->0
Node #4:
The Longest distance of it:0.000000
The Longest path of it:4->0
```

(5) Result demonstration

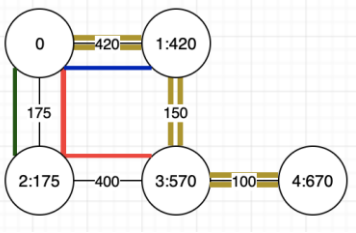
According to what we just entered into the program, we can draw out a graph:



For each node, we can see the results match the graph:

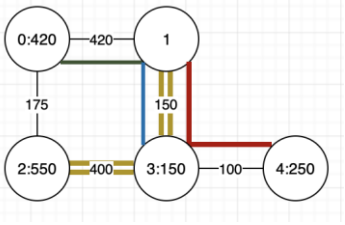
I. Node 0:

```
Path for 0:
Farthest Node #4:
Its Longest Distance:670.000000
Path: 0->1->3->4
0's distance: 0.000000
1's distance: 420.000000
2's distance: 175.000000
3's distance: 570.000000
4's distance: 670.000000
```



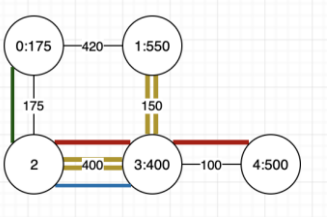
II. Node 1:

```
Path for 1:
Farthest Node #2:
Its Longest Distance:550.000000
Path: 1->3->2
0's distance: 420.000000
1's distance: 0.000000
2's distance: 550.000000
3's distance: 150.000000
4's distance: 250.000000
```



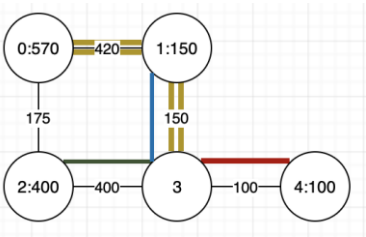
III. Node 2:

```
Path for 2:
Farthest Node #1:
Its Longest Distance:550.000000
Path: 2->3->1
0's distance: 175.000000
1's distance: 550.000000
2's distance: 0.000000
3's distance: 400.000000
4's distance: 500.000000
```



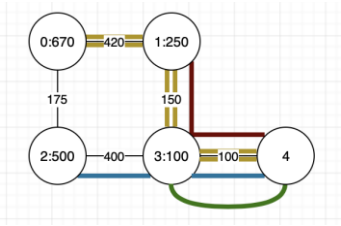
IV. Node 3:

```
Path for 3:
Farthest Node #0:
Its Longest Distance:570.000000
Path: 3->1->0
0's distance: 570.000000
1's distance: 150.000000
2's distance: 400.000000
3's distance: 0.000000
4's distance: 100.000000
```



V. Node 4:

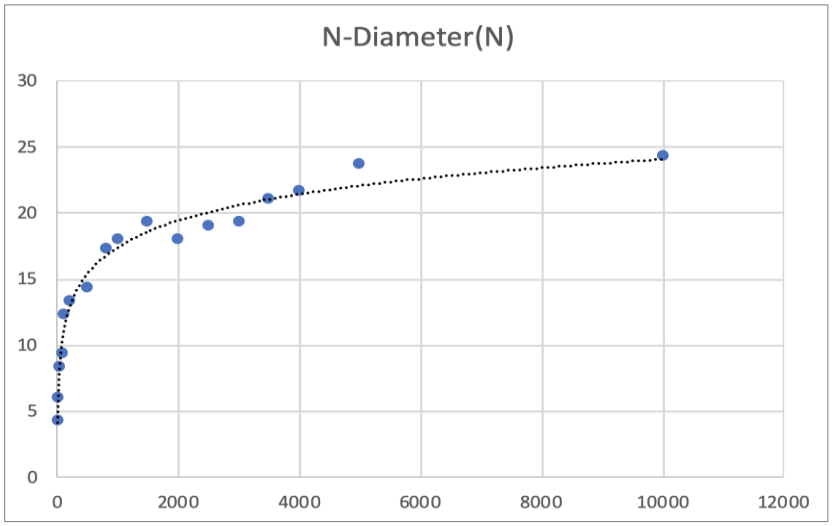
```
Path for 4:
Farthest Node #0:
Its Longest Distance:670.000000
Path: 4->0
0's distance: 670.000000
1's distance: 250.000000
2's distance: 500.000000
3's distance: 100.000000
4's distance: 0.000000
The diameter = 670.000000
```



As we can see, the results turn out to be consistent with the graph.

Part 4: Graph diameter analysis

After validating the correctness of the program, we utilize it to calculate the diameter of scale-free networks with different numbers of nodes from 10 to 10000 and get results shown in a N-Diameter(N) graph like this:



According to the analysis of scale-free networks from previous studies[2], the diameter of a graph with all the links having weight = 1 should be $\Theta(\log n)$. And the pattern shown in the graph of our results also fits this pattern.

References:

[1] Quartieri J, Guida M, Guarnaccia C, et al. Complex Network Applications to the Infrastructure Systems: the Italian Airport Network case[J]. New Aspects of Urban Planning and Transportation, 2008.

[2] Béla Bollobás and Oliver Riordan, The Diameter of a Scale-Free RandomGraph. Combinatorica, 2004. 24(1): p. 5-34.