CSE 101 - Introduction to Programming
Homework 3
October 2018

<u>BACKGROUND INFORMATION</u>

1. Introduction to Graphs

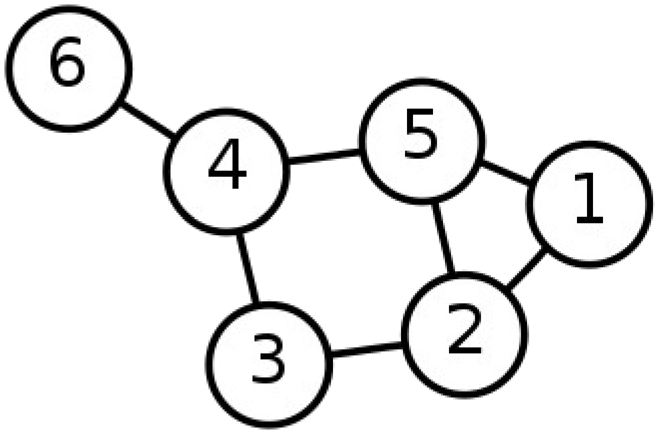
A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices or nodes, and the links that connect the vertices are called edges. The nodes may depict any form of data, the edges may have weight (these graphs are called weighted graphs) and the edges may also have a direction (these are called directed graphs). A graph is connected when there is a path between every pair of vertices. In a connected graph, there are no unreachable vertices.

Note: *For the scope of this assignment we will be working with graphs that are:*

- *Unweighted*
- *Undirected*
- *Connected*
- *Containing no loops or parallel edges*

Figure 1, is a graphic representation of the following unweighted, undirected, connected graph:
V = {1, 2, 3, 4, 5, 6}
E = {{1, 2}, {1, 5}, {2, 3}, {2, 5}, {3, 4}, {4, 5}, {4, 6}}
Here, V is the set of vertices and E is the set of edges.

Figure 1:



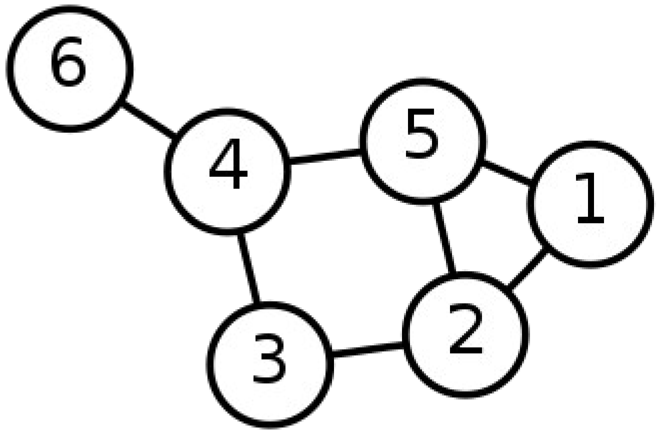
2. Application of Graphs

Graphs are used to represent many real life applications. Starting from networking to biology to operations research to e-commerce, graphs are used in many places. Graphs are also used in social networks like LinkedIn, Facebook, etc.

3. Concept of Shortest Path

For the scope of our discussion here, in the case of unweighted graphs the shortest path is the minimum number of hops or shortest length of the path required to reach the destination node from the source node. For every pair of nodes in a connected graph, there exists at least one shortest path between them. There may exist multiple shortest paths between a pair of nodes.

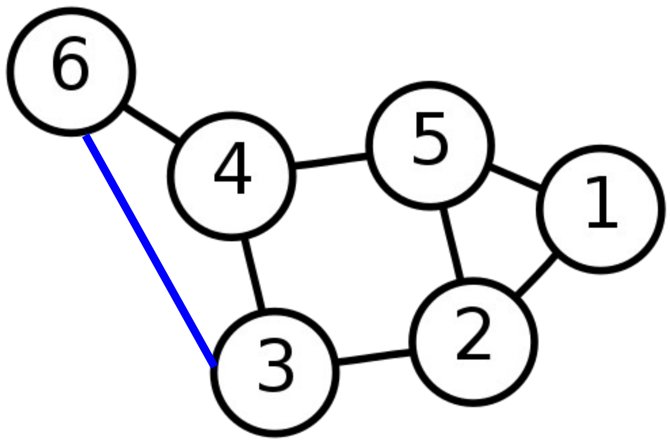
Figure 2:



In Figure 2, we can see that there can be multiple paths between node 1 and node 6. Some of them are:
1 - 5 - 4 - 6
1 - 5 - 2 - 3 - 4 - 6
1 - 2 - 3 - 4 - 6

The shortest path between 1 and 6 is 1 - 5 - 4 - 6.

Figure 3:



In Figure 3, we introduced a new edge {3,6}. In this case we can see that there are 2 same length shortest paths from 1 to 6: 1 - 5 - 4 - 6 and 1 - 2 - 3 - 6.

4. Betweenness Centrality

In layman terms, Betweenness Centrality is a metric that measures the importance of each node in a graph/network by giving it a numerical value. The nodes with the highest values of Betweenness Centrality will be the most important nodes in the graph.

The betweenness centrality of a node v is given by the expression:

$$g(v) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the total number of shortest paths from node $s$to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

Standardized Betweenness Centrality for undirected graphs with N nodes is calculated as the following:

$$Standardized\ g(v) = \frac{g(v)}{\frac{((N-1)(N-2))}{2}}$$

5. Applications of Betweenness Centrality

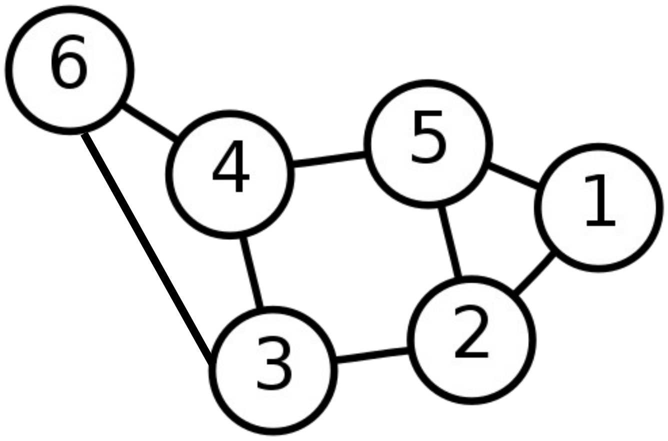
Betweenness centrality finds wide application in network theory apart from biology, transport and scientific cooperation: it represents the degree of which nodes stand between each other. For example, in a network, a node with higher betweenness centrality would have more control over the network, because more information will pass through that node. This is used extensively in social media analysis.

Suppose, you are given a network consisting of users and links between them.
Examples: Facebook-style networks: Nodes: people; Links: “friend”, messages
Twitter-style networks: Nodes: Entities/people Links: “follows”, “retweets”
The operations that can be done on such networks: What role does a node play in this network? Which are the most crucial nodes in linking densely connected regions of the graph? The crucial nodes can be found out using the Betweenness Centrality. Higher the Betweenness Centrality, more crucial the node becomes.

6. Example for Calculating Betweenness Centrality

Step 1. Choose the node for which the Betweenness Centrality needs to be calculated.
Step 2. Leaving aside this node make all the possible pairs with the nodes left.
Step 3. For each pair calculate the number of shortest paths (X) and the number of shortest paths which include our chosen node (Y).
Step 4. Against each pair calculate Y/X
Step 5. The sum of all these values will give us the Betweenness Centrality
Step 6. Calculate Standardized Betweenness Centrality

Figure 4:



Calculation of Betweenness Centrality for Node 3 in Figure 4:

Node Pairs (after excluding Node 3)	No. of shortest path between the node pair (X)	No. of shortest path between the node pair passing through Node 3 (Y)	Y/X
1,2	1	0	0
1,4	1	0	0
1,5	1	0	0
1,6	2	1	0.5
2,4	2	1	0.5
2,5	1	0	0
2,6	1	1	1
4,5	1	0	0
4,6	1	0	0
5,6	2	1	0.5
Betweenness Centrality for Node 3			2.5
Standardized Betweenness Centrality for Node 3			2.5/(((6-1)(6-2))/2) =2.5/10 =0.25

Similarly, Betweenness Centrality for all nodes can be calculated.
What can you say about the Betweenness Centrality for the other nodes in Figure 4 intuitively?

<u>**TASK**</u>

Given a graph to report top-'k' nodes which have exact largest Standardized Betweenness Centrality score.

If there is a list of nodes sorted in decreasing order according to their Standardized Betweenness Centralities, the first ‘k’ nodes with equal Standardized Betweenness Centrality from this list will be the output.

For accomplishing this last we can divide the task into sub-tasks:

1. Creation of the graph

2. Calculating the Standardized Betweenness Centrality of each of the nodes
    a. Calculating no. of shortest paths between a pair of nodes (say *A* and *B*)
        i. For this we will use a modification of the Depth First Search (DFS).
        ii. First, find any one shortest path between *A* and *B* and calculate it’s length (say *dist*).
        iii. Now, we will find all paths which are at a distance of *dist* from node *A.*
        iv. The paths which end at *B* will be our required set of paths. Let the no. of these paths be *X.*

    b. Calculating no. of shortest paths between a pair of nodes passing through the node chosen (say *C*) for calculating the Standardized Betweenness Centrality
        i. Out of all the shortest paths between *A* and *B,* we calculate the number of paths passing through node *C*.
        ii. Let the no. of shortest paths passing through *C* be *Y.*
        iii. We can easily get the value of *X/Y*

    c. Repeat steps a and b for each pair of nodes. The summation of all the *X/Y* will be the Betweenness Centrality of node C. From this we can also calculate the Standardized Betweenness Centrality.

    d. Similarly, repeat steps a,b and c to get the Standardized Betweenness Centrality of all the nodes in the graph.

3. Producing the output of top-‘k’ nodes

PARTIAL PSEUDOCODE FOR STEP 2 OF THE TASK

You will have to look at the starter.py file uploaded along with this file.

```
#Finds minimum distance between start_node and end_node; Implement BFS for shortest path length
function min_dist(start_node, end_node)
```

```
#Finds all paths from node to destination with length = dist
function all_paths(node, destination, dist, path)
    add node to path

    if path.length == dist
        if node == destination
            return path
        else
            return None

    my_paths = []

    for next_node in neighbours(node)
        if next_node not in path
            returned_paths = all_paths(next_node, destination, dist, visited, path)

            if returned_paths is not None
                add returned_paths to my_paths
    if my_paths is not empty
        return my_paths
    else
        return None

#Finds all shortest paths between start_node and end_node
function all_shortest_paths(start_node, end_node)
    dist = min_dist(start_node, end_node)
    paths = all_paths(node=start_node, dest=end_node, dist, visited={}, path=[])
    return paths
```

GENERAL INSTRUCTIONS

1. You may use any basic data structures that are available in Python, e.g., list, tuple, struct, dictionary, set, etc.
2. As a part of this assignment you need to create the code for finding out the shortest path.
3. You are only allowed to use the basic modules available in Python. You cannot use any external modules, libraries or APIs.

4. Please create a document citing all the online and offline resources that you used in doing this assignment.
5. Your code will be checked for plagiarism against code of your classmates as well as some sample codes available online.

SUBMISSION INSTRUCTIONS

1. Your main code file must be renamed SBC_2018xxx.py
2. Your document with references to resources used must be in pdf format and named SBC_Resources_2018xxx.pdf
3. These must be put in a zipped folder named HW3_2018xxx.zip