Beihang University
School of General Engineering
Beijing, Dec. 13th, 2019
Sebastian Wandelt, Junyu Zhao, Pengfei Yi

Computer Science and Programming
Lab Class 13

Task 1. ***Graph – class*** *(20 minutes)*

Implement a graph class to describe a given graph with the information from the adjacency list of a graph. Use Figure 1 as an example testing the class. In specifc, one should first construct the adjacency list of the graph shown in Figure 1, and use the functions to add nodes and edges in the class. That is, the class should contain a constructor, a function for adding a node, a function for adding an edge.

1. __init__(self):
2. add_node(self,node):
3. add_edge(self,node1,node2): #Here edge=(node1,node2)

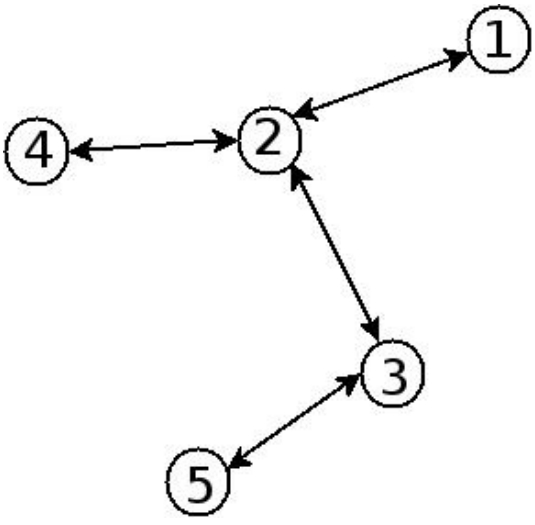

Figure 1: The graph example for the lab class.

Task 2. ***Graph – Properties*** *(20 minutes)*

Implement the following functions based on the graph class implemented in Task 1:

1. Get all nodes.
2. Get all edges.
3. Get the degree of a given node in the graph.
4. Get all neighbor nodes of a given node.

Task 3. ***Networkx – Graph construction and properties*** *(30 minutes)*

To more conveniently access the properties of a given graph and perform operations with the graph, the package networkx is introduced.

1. Use networkx package to construct the graph shown in Figure 1. One could refer to the documents in the official website of networkx: `http://networkx.github.io/`.
2. Implement the four functions in task 2. Compare the results from networkx with the functions implemented by your class.
3. Get the shortest paths between all pairs of nodes using nx.shortest_path(G).
4. Get the betweenness centrality of each node.

Task 4. ***Networkx – Depth-first search*** *(20 minutes)*

Implement the depth-first search (DFS) with the graph built with networkx in Task 3. The function should show the order accessing each node. Think about the computational complexity of DFS.

Task 5. ***Networkx – Breadth-first search*** *(20 minutes)*

Implement the breadth-first search (BFS) with the graph built with networkx in Task 3. The function should show the order accessing each node. Think about the computational complexity of DFS.