Beihang University
School of General Engineering

Beijing, March 2nd, 2021
Lin Wei and Yida Ding

Data Management and Artificial Intelligence
Lab Class 1

Task 1 ***Graph – Breadth-first search****(15 minutes)*

Implement the breadth-first search (BFS) with the graph in homework. The function should show the nodes at each breadth. Think about the time complexity of BFS.

Task 2 ***Graph – Dijkstra's Algorithm*** *(20 minutes)*

Please construct an undirected graph visualized in Figure 1 using Networkx. Note to assign weights to edges when constructing graph. Then, implement the Dijkstra's algorithm to obtain the lengths of shortest paths from node A to the other nodes. The desired output should be: $\{A:0, B:7, C:3, D:9, E:5\}$

Task 3 ***Graph – Dijkstra's Algorithm to obtain paths*** *(30 minutes)*

Based on Dijkstra's Algorithm, please obtain the shortest path from node A to the other nodes. The desired output should be: $\{A:[A], B:[A,C,B], C:[A,C], D:[A,C,B,D], E:[A,C,E]\}$

Hint: Create a dictionary `parent` to record the parent of each node in shortest path tree. Whenever we find shorter path through a node u, we make u as parent of current node. Once we have parent dictionary constructed, we can obtain path using recursive function.

Task 4 ***Graph – Kruskal's Algorithm*** *(30 minutes)*

Implement the Kruskal's algorithm in Python with the graph visualized in Figure 2. For convenience, you can start with the code of graph construction below. The desired output should be: $\{(F,G),(A,B),(C,F),(A,D),(E,F),(A,E)\}$

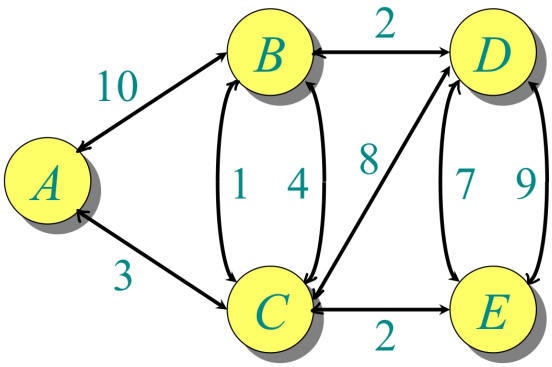

Figure 1: Graph of Task 2 and 3

```
G=nx.Graph()
G.add_edge('A','B',weight=2)
G.add_edge('A','D',weight=8)
G.add_edge('A','E',weight=14)
G.add_edge('D','E',weight=21)
G.add_edge('B','E',weight=25)
G.add_edge('B','C',weight=19)
G.add_edge('E','C',weight=17)
G.add_edge('E','F',weight=13)
G.add_edge('C','F',weight=5)
G.add_edge('C','G',weight=9)
G.add_edge('F','G',weight=1)
```

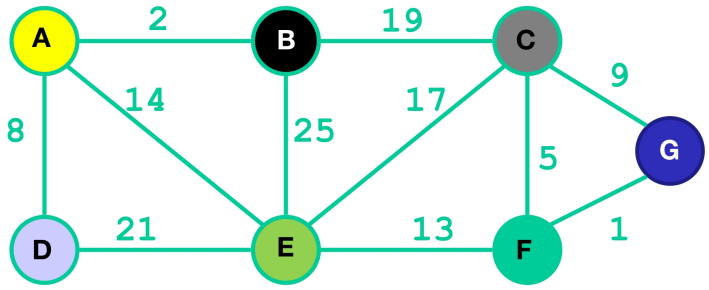

Figure 2: Graph of Task 4