1) – Explanation of 1st question code

The code begins by importing the necessary libraries: itertools for generating combinations and networkx for graph manipulation and analysis.

The generate\_connected\_subgraphs function takes an integer n as input, which represents the number of nodes in the complete graph.

Then, Inside the function, an empty directed graph (DiGraph) is created using the nx.complete\_graph function. This graph initially contains all possible edges.

All possible edges are extracted from the graph using graph.edges() and stored in the all\_edges list.

The code then generates all possible combinations of edges with lengths ranging from 1 to the total number of edges.

Each combination of edges is added to the combination list.

Next, the code iterates over each combination and creates a new subgraph using nx.DiGraph(). The selected edges from the combination are added to the subgraph using subgraph.add\_edges\_from(combo). The code checks if the subgraph is weakly connected (all nodes are reachable) and has the desired number of nodes (n). If both conditions are met, the subgraph is appended to the subgraphs list.

After generating all possible subgraphs, the code checks for isomorphism between the subgraphs. Only unique subgraphs are kept in the unique\_graphs list, discarding any isomorphic duplicates.

Finally, the function returns the list of unique subgraphs.

The code then proceeds to iterate over values of n from 1 to 7 (inclusive) using the range function.

If n is 1, it writes "no edges" to a file named "connected\_sub-graphs.txt". Otherwise, it calls the generate\_connected\_subgraphs function to obtain the subgraphs for the current n.

The obtained subgraphs are then written to the "connected\_sub-graphs.txt" file, along with information about the count of subgraphs and the edges in each subgraph.

The execution time for generating subgraphs for the current n is printed, and the start time is updated for the next iteration.

Overall, the code systematically generates all possible connected subgraphs of a complete directed graph, filters out isomorphic duplicates, and saves the results to a text file for analysis.

2) – Explanation of 2st question code

The code starts by importing the necessary libraries: itertools for generating combinations and networkx for graph manipulation and analysis. It also imports the sys module for handling n=1 case.

The user is prompted to enter a positive integer n.

If n is 1, the program exits with an error message stating that the input is invalid.

Else, A directed graph G is created (the given graph in the question) using the nx.DiGraph() function, and a few example edges are added to the graph.

All edges of the graph are extracted from the graph using G.edges() and stored in the all\_edges list.

The code initializes empty lists to store subgraphs and combinations of edges.

The code then generates all possible combinations of edges of the graph G, with lengths ranging from 1 to the total number of edges using nested loops. Each combination of edges is added to the combination list.

The code iterates over each combination and creates a new subgraph using nx.DiGraph(). The selected edges from the combination are added to the subgraph using subgraph.add\_edges\_from(combo).

The code checks if the subgraph has the desired number of nodes (n). If the condition is met, the subgraph is appended to the subgraphs list.

After generating all possible subgraphs, the code checks for isomorphism between the subgraphs. A dictionary called motifes is initialized with the first subgraph as the key and an empty list as the value. The dictionary is used to group isomorphic subgraphs together.

For each subgraph g in the subgraphs list, the code checks if it is isomorphic to any existing key in the motifes dictionary. If it is, g is added to the list of corresponding key. If it is not, a new key is added to the dictionary with g as the value list.

The code then proceeds to write the results to a text file named "sub-graphs\_motifes.txt".

If n is 1, it writes "no edges" to the file. Otherwise, it writes the value of n and the count of subgraphs.

For each motif (isomorphic group of subgraphs) in the motifes dictionary, the code writes the count of subgraphs in that motif and the edges of the first subgraph in the motif. Finally, the file is closed.

Overall, the code generates all possible subgraphs of a directed graph and groups them based on isomorphism. The results are then saved to a text file for further analysis.