BT-3051 Data Structures and Algorithms for Biology
Assignment - 6

Submission : Since this is a coding based assignment, students need to submit their codes in a zipped folder. Your zip file should be named something like BTyyBxxx.zip, based on your roll number. This zip file must contain the codes used for each of the problems in separate **.ipynb** files with proper documentation.

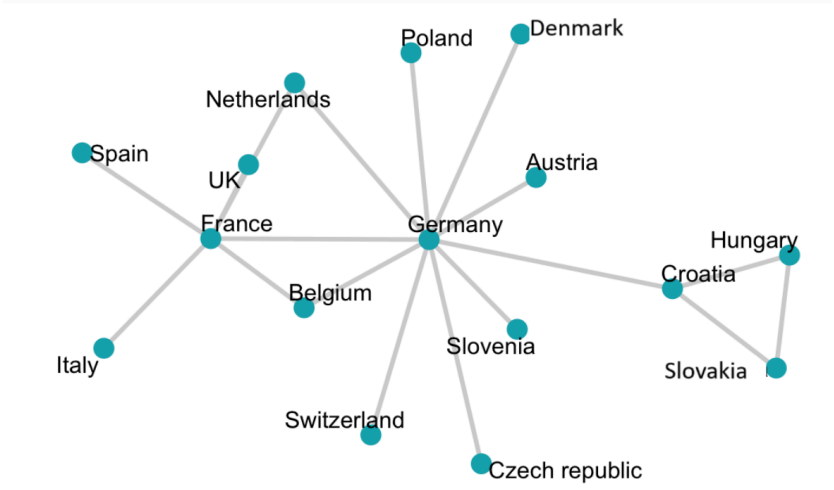
Deadline : 4th November 2024, 23:59 hrs
Bonus marks and penalty will be applied as mentioned in the course plan.

Instructions:

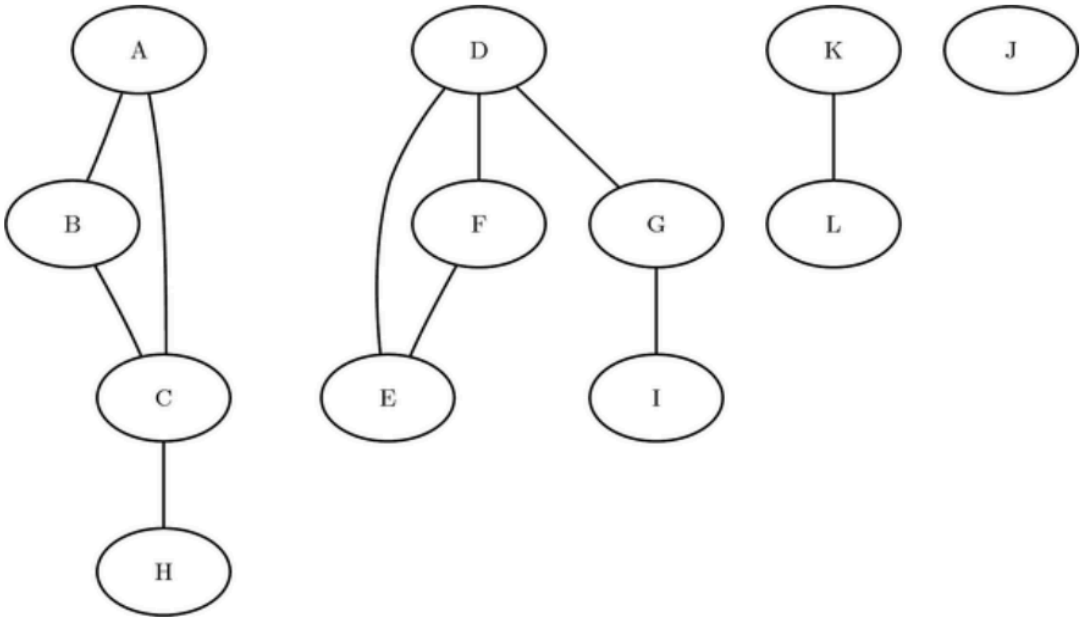
a. Total marks for this assignment: 30. Each question carries 5 marks.
b. No restrictions on usage of functions, unless otherwise specified.
c. Submissions will be evaluated only if they are provided in the required .ipynb format.

Q1. Breadth-First Search (BFS) and Depth-First-Search (DFS) are algorithms used for graph traversal. Implement these two algorithms with an additional task of constructing the traversal tree generated by both algorithms. A traversal tree captures the order in which nodes or vertices are visited during the traversal. Were you able to identify all the edges in the original graph as well as in the traversal tree? If not, please provide insights into any missing edges and their implications within the graph. Perform this on the graph as illustrated Below.

Starting node: Germany
Traversing order: Alphabetical priority (A-Z)
Your code should return traversal tree(both BFS,DFS) as an object of networkx and visualize it.



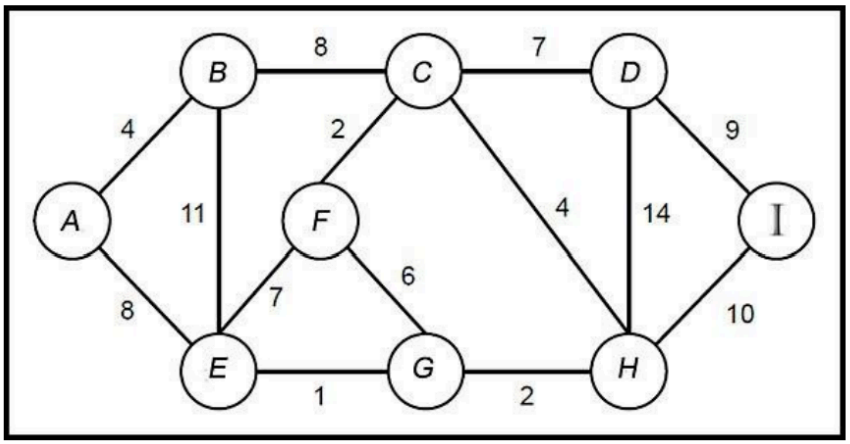
Q2. Connected components are subsets of a graph in which every node is reachable from every other node in that subset by following edges. These components often represent distinct groups or communities within a graph. Your task is to repurpose Depth-First Search (DFS) or Breadth-First Search (BFS) to determine the number of connected components in any given graph. Your code should print the number of connected components and return a dictionary with component id as key and list of nodes as values



Q3. Emergency Response Planning

You are responsible for planning emergency response routes for a city. The city is represented as a graph, with intersections as nodes and roads as edges. In the event of an emergency, you need to find the shortest path from a central emergency command center to various locations in the city to minimize response time. The city's road network is represented as an undirected weighted graph, where each road has a known travel time (weight) associated with it. Your task is to find the shortest paths from the central emergency command center (Node F) to all other intersections in the city.

Your code should return a nested list of [Source, Target, Duration]. Ex [[F,C,2],....]



Q4. You are provided with a text file (***Q4.txt***) containing a scene from the TV show *The Simpsons*. The task is to extract all the sentences spoken by Bart from the text using regular expressions. Construct a suffix trie using the words in each sentence spoken by Bart, where each word is a node in the trie. a) Write a function that checks if a given sentence was spoken by Bart by searching the suffix trie. b) Write another function that autocompletes a partially entered sentence by returning potential completions from the trie based on sentences spoken by Bart.

Example input and output:

a) "I want to read Hamlet" - False

"I want to play cricket" - False

"My whole life is a lie" - True

b) "My name is" : "My name is Lance Ambu", "My name is Bart"

Q5.

a) Use a single regular expression to tokenize the given SMILES (Simplified Molecular Input Line Entry System) strings representing chemical compounds into atoms, bonds and other symbols. Terms within square brackets need to be kept as is. The SMILES strings and their expected tokens are given.

SMILES	**Tokens**
N[C@@H](C)C(=O)O	'N', '[C@@H]', '(', 'C', ')' , 'C', '(', '=', 'O', ')', 'O'
N#N	'N', '#', 'N'
[Cu+2].[O-]S(=O)(=O)[O-]	'[Cu+2]', '.', '[O-]', 'S', '(', '=', 'O', ')', '(', '=', 'O', ')', '[O-]'
O1C=C[C@H]	'O', '1', 'C', '=', 'C', '[C@H]'

b) Given a SMILES string, write a function to check if it is a valid one. The rules here are:

- The following are valid symbols for atoms: **`C, N, O, F, P, S, Cl, Br, I`**, and their lowercase equivalents.
- Bonds can be described by **=**, **#**, and **.**
- Branches represented by **()** should be balanced (every '**(**' should have a matching '**)**')
- **[]** is allowed. Any other character apart from the given ones are not allowed.

Q6. A circular plasmid of size ~100 bases was sequenced, and overlapping short reads of size 3 (3-mers) were generated. The short reads are provided in the file ***`Q6.txt`***.

Construct a de Bruijn graph from the short reads and **reconstruct the original plasmid sequence** by finding an Eulerian circuit that represents a valid traversal through all the k-mers.

a) Write a function that takes a list of short reads as input and returns the reconstructed genome. If an Eulerian circuit cannot be found, the function should return *`None`*.
b) Visualize the de Bruijn graph using the `networkx` library.
c) Compare the number of nodes and edges in the graph with the number of reads and comment on your observation.