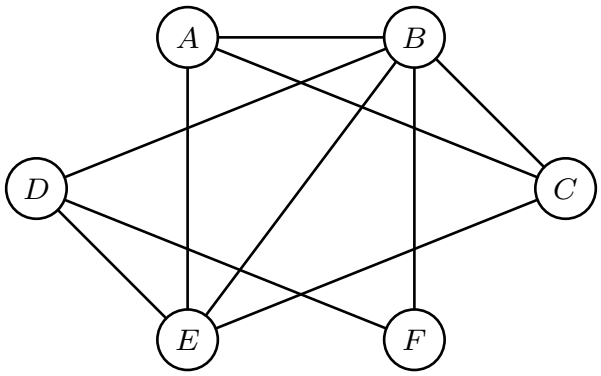
GT Lab Practical 01-Introduction to Graphs Theory

Lecturer: Pheak NEANG, Vinha CHHAENG

October 16, 2025

1. Undirected Graph: Social Media Network

Given a graph $G(V, E)$, where V denotes the set of vertices and E denotes the set of edges, a connection between two vertices without direction is referred to as an **undirected graph**. In this part, we will simulate a friendship network on the Facebook platform using the `NetworkX` library in Python. The graph $G(V, E)$ will represent the social network, where each vertex corresponds to a user account (e.g., $A, B, C, \ldots$), and each edge represents a friendship connection between two users (e.g., $(A, B), (A, E), \ldots$) as illustrated in the graph below.



Question 1. Graph Simulation in Python using `NetworkX` packages.

(1.1). Recall the network G by setting its name as `graph` using the function `nx.Graph()` to create an undirected graph.

(1.2). Create two lists, `ver` and `arc`, and a dictionary named `pos` to store all elements of the vertex set V, the edge set E, and the positions of the vertices in the graph network G, respectively.

(1.3). Build the graph network $G(V, E)$ by adding the vertices V and edges E obtained from variable `ver` and `arc` using the functions `graph.add_nodes_from()` and `graph.add_edges_from()`.

(1.4). To visualize the graph, first draw the nodes using the function `nx.draw_networkx_nodes()`, then add labels to the nodes with `nx.draw_networkx_labels()` then draw the edges using `nx.draw_networkx_edges()` and finally `plt.show()` to display this graph.

Question 2. Some definition and graph properties.

(2.1). Let $\Gamma(v)$ denote the set of neighbors of a vertex v in the graph G. Obtain the list results of $\Gamma(A)$ and $\Gamma(E)$ using the function `graph.neighbors()`. Then, plot the neighbors of vertex A in red and the neighbors of vertex E in blue.

(2.2). Recall that the function `graph.degree(v)` represents the degree of a node or vertex v in the graph G in `NetworkX`. Write a function named `show_connection()` to display the nodes that have the smallest and largest numbers of connections in this social network. Determine which user is the **introvert** (having the fewest connections) and which is the **extrovert** (having the most connections) in this network.

(2.3). Let $\Lambda(v, w)$ be the set of vertices that form connections from node v to node w in the graph G. Recall that the function `all_simple_paths(graph,v,w)` in `NetworkX` can be used to obtain all possible paths between v and w. Write a function named `smartest_path()` to determine the path that involves the minimum number of user accounts between two nodes. Test your function to find how many intermediate users are required to reach user E starting from user F then add green colors on edges that show a path from F to E.

Question 3. User Account has been Hacked and new user is created.

(3.1). Assume that the user account C has been hacked, alert this notification by adding rad color on user C and all connection to another users.

(3.2). All user accounts that are friends with user C have been blocked. Therefore, you need to unfriend user C and remove this user from the network. Use the function `graph.remove_edge(u, v)` to disconnect the edge between vertices u and v, and `graph.remove_node(u)` to remove vertex u from the graph G. Finally, draw the updated graph network G without vertex C.

(3.3). Three new user accounts (Z, J, K) have now been created, and user C has been renamed to C_0. The social friendship network has been updated according to the table listed below:

User	Network
Z	A, D, J
J	A, B, F
K	D, E, Z
C_0	A, B, E

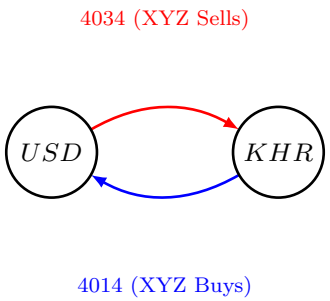
Adding these new user accounts using the function `graph.add_node(v)` to add a vertex v to the graph G, and creating the corresponding edges to update the network in graph G. Finally, plot the updated graph.

2. Directed Graph: Currency Exchange Network

Setting the directed graph $G(V, E, W)$ to represent the network of currency exchanges, where V is the set of currencies, E is the set of exchange relationships between currencies, and W is the set of exchange rates associated with each pair of connected currencies. To begin, the exchange rates of several currencies USD, EUR, KHR,AUD, and JPY at Bank XYZ on 16 October 2025 are provided in the table below.

Currency	XYZ Buys	XYZ Sells
USD/KHR	4014	4034
EUR/USD	1.125558	1.181335
JPY/KHR	25.1539	27.0393
USD/JPY	148.450589	160.372745
AUD/USD	0.616106	0.658097
AUD/KHR	2485.37	2641.60

Each exchange rate represents the value of one unit of the foreign currency in comparison to one US dollar (USD). For example, the exchange from USD to KHR at Bank XYZ is represented by two columns: **XYZ Sells** at 4034, meaning that for $1, USD$, the bank sells it at this rate; and **XYZ Buys** at 4014, meaning that for $1, USD$, the bank buys it at this rate and graph $G(V, E, W)$ given by :



Question 1. Model this problem as a graph and then simulate it in Python using the `NetworkX` package.

(1.1). Setting the graph of exchange rates $G(V, E, W)$ as `G`, and creating this graph using the function `nx.DiGraph()`.

(1.2). Define two lists, `node` and `edge_var`, and a dictionary named `pos` to store all elements of the vertex set V, the edge with weight set (E, W) , and the positions of the vertices in the graph network G, respectively.

(1.3). Build the graph network $G(V, E, W)$ by adding the vertices V and weight of edges (E, W) obtained from variable `node` and `edge_var` using the functions `graph.add_nodes_from()` and `graph.add_weighted_edges_from()`.

(1.4). Given an example of the exchange rate where Bank XYZ *Sells* between USD and JPY, and Bank XYZ *Buys* between EUR and USD, using the function `nx.get_edge_attributes()`.

(1.5). To visualize the graph, first draw the nodes using the function `nx.draw_networkx_nodes()`. Next, add labels to the nodes with `nx.draw_networkx_labels()`. Then, draw the edges using `nx.draw_networkx_edges()` with an additional attribute called `connectionstyle` to modify the shape of the edges. Finally, use `plt.show()` to display the graph.

Question 2. Some definition and graph properties.

(2.1). Define $\Gamma(v)$, $\Gamma^+(v)$, and $\Gamma^-(v)$ as the sets of neighbors, successors, and predecessors of a vertex v in graph G, respectively. Recall the corresponding functions in `NetworkX`: `G.neighbors()`, `G.successors()`, and `G.predecessors()`. Determine the sets $\Gamma(AUD)$, $\Gamma^+(USD)$, and $\Gamma^-(EUR)$.

(2.2). Suppose you hold a currency in JPY. Identify the currencies with which it can be exchanged.

(2.3). Let $\gamma^+(u, v)$ is a set of vertices following direction from u to v in graph G. Recall that the function `all_simple_paths(graph,u,v)` in `NetworkX` can be used to obtain all possible paths between u and v. Find possible path of currencies network from JPY to KHR.

(2.4). Is it possible, in the case where you hold a currency in EUR, to exchange it into KHR? If so, how can this exchange process be represented and described the network in graph model?

(2.5). Let $\phi^{+}(u)$ be a set of vertices following direction from u to u self in graph G was called cycle. Find the circle of vertex USD by using function `nx.simple_cycles()`.

Question 3. Basic network analysis of exchange rates at Bank XYZ.

(3.1). Suppose you have 1000 (USD) and sequentially exchange it into other currencies before converting it back to your original currency (USD). Would this series of exchanges result in a profit? If so, calculate the total amount you would receive.

(3.2). Now assume that you have 1000 (EUR). The objective is to determine the most profitable way to exchange it into (KHR).