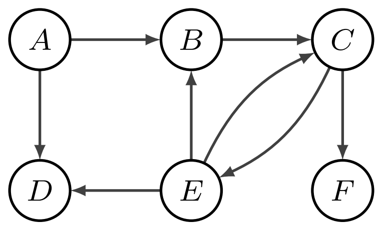
GT Lab Practical 03 - Strongly Connected Components in Graph

Lecturer: Pheak NEANG, Vinha CHHAENG

November 2, 2025

1. Definition of connectivity in graph

Given that a graph **G(X,U)** where **X** and **U** be the sets of nodes and edges in graph is shown in figure below:



Question 1. Strongly Connected Components

(1.1). Applied **DFS** algorithm to find strongly connected components (SCC) with the previous graph starting from a node of your choice.Your result should be an order list of sets where each set contain the vertices in SCC.

(1.2). For each SCC list in **(1.1)**, determine which are sink components, which are source components, and which are belong to neither category.

(1.3). Determine the minimum number of of edges that must be added to make the graph strongly connected (i.e. the graph has only one SCC).Justify your answer by providing the edges required.

(1.4). Write and execute Python code for connected components on the graph **G** to be verify your result.

Question 2. Find connected components (CC) by remove the direction and edges $[(E, C), (C, E), (B, C)]$ in graph **G**.

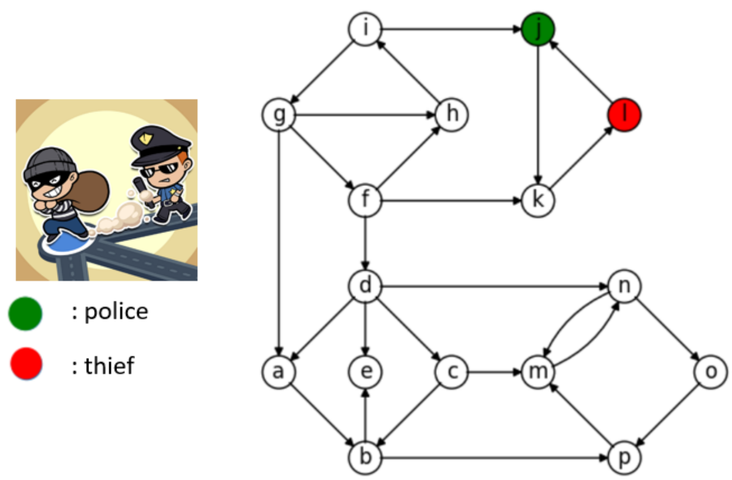
(2.1). Run the connected components and **DFS** algorithm with the previous graph starting from a node of your choice. List the order of each node in sets of connected components (CC). How many CC for graph **G** ?

(2.2). Determine the minimum number of of edges that must be added to make **G** is connected graph.

(2.3). Write and execute python code for connected components algorithms on the graph **G**. Present the code and the output, demonstrating the traversal from a selected starting node.

2. Problem: Catch The Thief

One day, a thief committed a crime. He known for his cunning and stealth, successfully robbed a large of money from the central bank. The police, determined to bring the criminal to justice, have launched a massive investigation.They have narrowed down the location of the thief, but with the complex layout of the city, finding and capturing him will not be easy. Given that city's layout is represented as a directed graph, with nodes symbolizing various locations and edges showing the connections between them. The green node represents the police's position, and the red node represents the thief's location is shown in figure below:



Condition:

- Police Movement: The police can move to any adjacent node that is directly connected to their current position.
- Thief Movement: The thief can also move to any adjacent node, trying to evade capture.

Objective: The goal for the police is to plan their movements to arrest the thief. To ensure success, the police need to positioning the thief in such a way that the police can catch him. Develop a strategy based on the graph's structure and start preparing your plan.

Q1. In which nodes of the graph G_0 are able to assign thief's position so that the police can catch him? List all possible results.

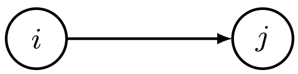
Q2. Now, by adding a directed edge from node(e) to node(d) noted graph G_1, is it still possible for the police to catch the thief? Justify your answer.

Q3. If not, how many assistants are needed to help the police at a minimum? Assign the positions for each assistant on graph G_1 and draw the graph to show your final solution.

3. Problem: Social Network Group Analysis

In this section, we introduce the concept of strongly connected components in graph theory to analyze a social network represented by a relationship matrix denoted as `known`. The objective is to determine the number of groups formed based on the notions of friendship and true friendship.

The matrix `known[i][j]` defines the relationship between two individuals, where it indicates whether person `i` knows person `j` according to the following conditions:



`known[i][j] = "Y"`

`known[i][j] = "N"`

- `known[i][j] = "Y"` indicates that person `i` knows person `j`.
- `known[i][j] = "N"` indicates that person `i` does not know person `j`.

Based on this relationship matrix, the following definitions are applied:

- A person `i` is considered a **friend** of `j` if `j` knows `i`, or if `j` has a friend who knows `i` either directly or indirectly.
- Two people `i` and `j` are **true friends** if `i` is a friend of `j` and `j` is a friend of `i`.
- A **group** is defined as a set of individuals in which every member is a true friend of every other member.

For testing purposes, consider an array of strings representing the relationship matrix `known[i][j]` of size $N \times N$, where N denotes the total number of people and $i, j \in [0, N]$.
As an example, let us consider a network of $N = 7$ individuals whose relationships are defined by the following matrix:

$$\texttt{known[i][j]} = \begin{bmatrix} N & N & N & N & Y & Y & N \\ N & N & N & N & N & N & N \\ N & N & N & N & N & Y & N \\ N & Y & N & N & N & Y & Y \\ N & N & N & Y & N & N & N \\ N & N & Y & N & N & N & Y \\ N & N & N & N & Y & N & N \end{bmatrix}$$

Q1. Graph modeling of the friend network in social media by named the seven people by $[A, B, C, D, E, F, G]$ and the relationship defind by matrix `known[i][j]`.

Q2. Write your own code to determine the number of groups that can be formed from the given relationships true friend. (Answer: 3 group)