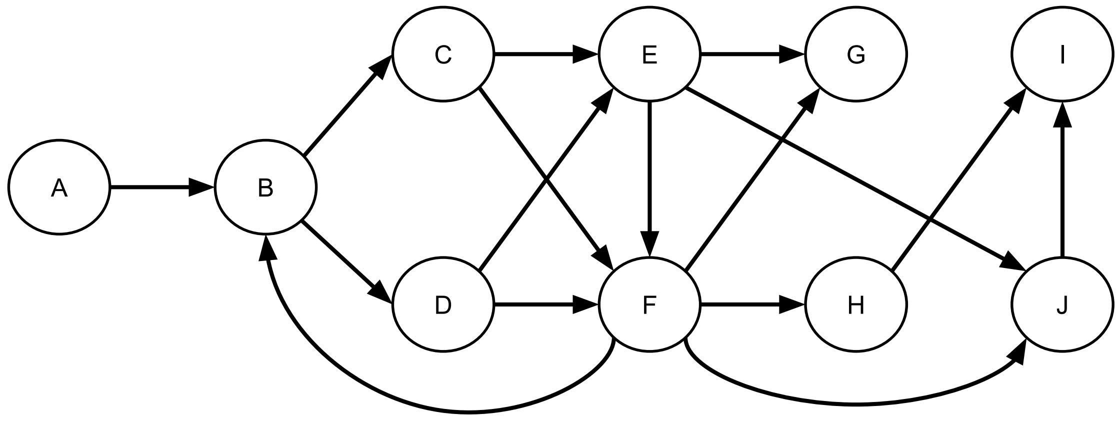
1 Problem Basic Graph

a) Given an adjacency matrix 1. Your tasks are as follows:

1. Convert the adjacency matrix to an adjacency list. The adjacent lists need to be in alphabetical order.
2. Draw the adjacency matrix in graph form.
3. Make an adjacent list out of the graph in Figure 1 which also needs to be in alphabetical order.

```
#         1  2  3  4  5  6
Adj = [[0, 1, 0, 0, 0, 0], # 1
       [0, 1, 1, 1, 0, 0], # 2
       [1, 1, 0, 0, 1, 0], # 3
       [0, 0, 0, 0, 1, 1], # 4
       [0, 0, 1, 1, 0, 0], # 5
       [0, 0, 0, 1, 0, 0]] # 6
```

b) Given a graph $G = (V, E)$ and a source vertex s, execute both Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms. Show the process by doing it manually on the graph depicted in Figure 1 with source vertex A (you can also do it with code, but then you need to include screenshots that clearly show the process). When visiting neighbors, visit them in alphabetical order. Each edge has a weight of 1. Don't forget to incorporate the start and finish times.

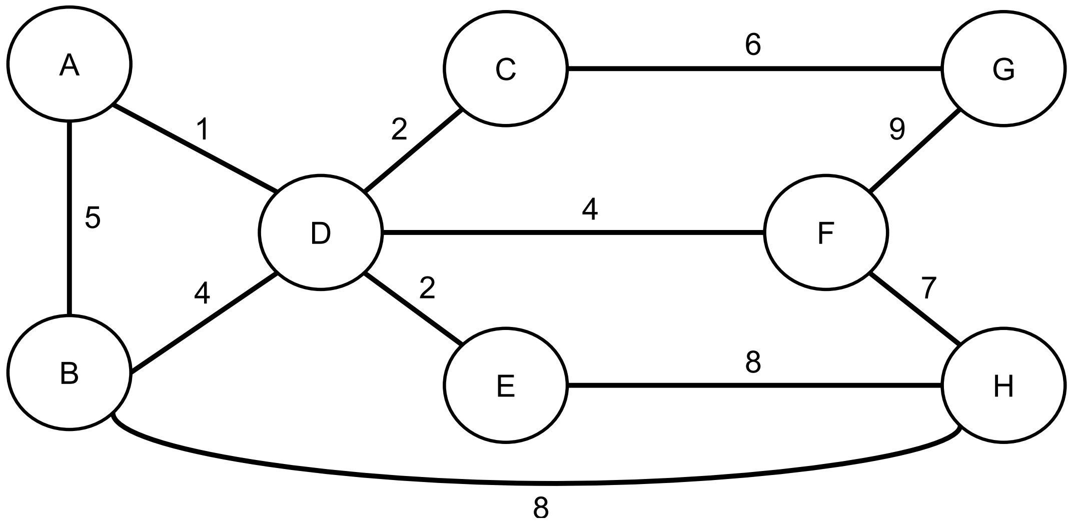


c) Remove a single edge from the graph $G = (V, E)$ in Figure 1 such that it becomes a DAG. Perform a topological sort on the resulting DAG.

d) Provide an algorithm capable of transforming a directed graph $G = (V, E)$ into a DAG. Test the algorithm on the graph shown in Figure 1, considering the presence of two additional edges: one from $I \to C$ and another from $C \to A$

2 Problem Cable Network

a) A city is intending to create a cable network to link all its neighborhoods. Each neighborhood corresponds to a vertex in the graph; the cables represent the edges. Each cable is assigned a specific cost. The city has set a budget limit, denoted as b. Your goal is to find out if it is possible to connect all the nodes in the network while still staying within the budget constraint. Your goal is to find out if it is possible on the undirected graph $G = (V, E)$ shown in Figure 2 with a constraint $b = 30$.



b) The vertex D is restricted to having a maximum of 3 edges. Is it still possible to meet the budget constraint using a modified algorithm from task a)? Will this solution always create a globally optimal solution? If not, show a counter-example.

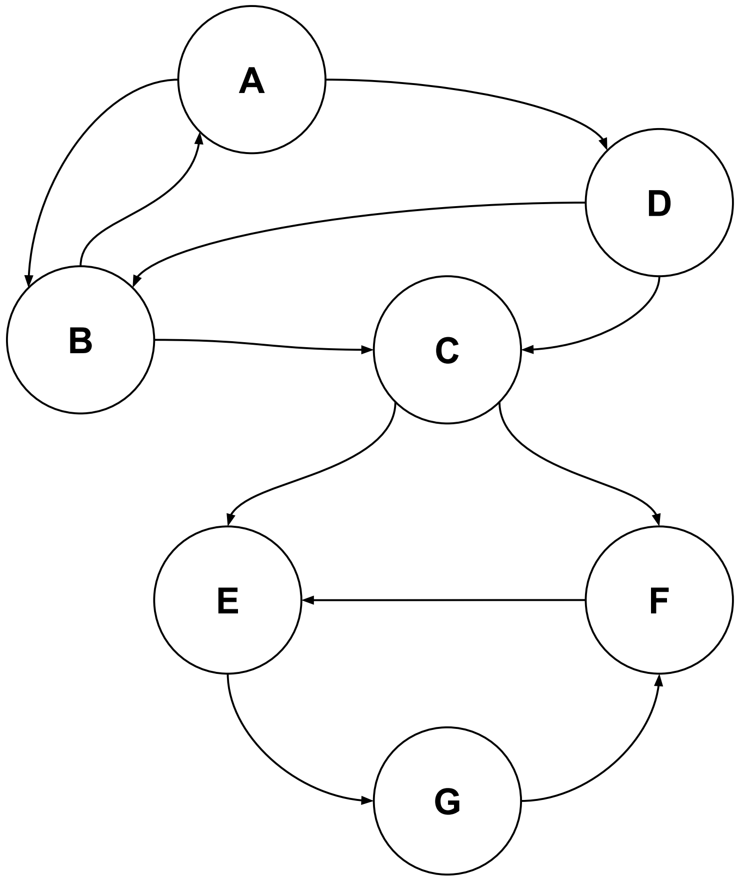
c) With a newly introduced constraint b', can we adjust the position of a single edge in the graph to make it compliant with this constraint? Show if it is possible on the undirected graph $G = (V, E)$ in Figure 2 with constraint $b' = 25$. An instance of modifying the position of a single edge involves swapping edge (A, B) with edge (C, G). This would result in the edge weight of $(A, B) = 6$, while the weight of edge $(C, G) = 5$.

3 Problem Finding Champion

Consider a directed graph $G = (V, E)$ where each edge (u, v) signifies that node u has defeated node v. Additionally, a node is considered to have defeated another node indirectly if there is a path between them. For instance, if a path from u to v' exists, we also define that u has defeated v'. A champion is a node that has defeated all the other nodes either directly or indirectly. In Figure 3, the champions are A, B, and D.

a) Your task is to propose an algorithm to identify and list all the champions in the directed graph, if there are any champions at all.

b) Rather than searching for champions, the goal is to divide the graph into groups, where each group comprises nodes that have defeated each other, either directly or indirectly. Present a solution for this subtask and indicate its running time. From Figure 3, the groups are divided into $\{A, B, D\}$, $\{C\}$ and $\{E, F, G\}$.



4 Problem Shortest Path

Suppose you have a directed graph $G = (V, E)$ where each edge (u, v) is assigned a weight, which may be positive, zero, or negative. The objective is to find an example of such a graph with arbitrary edge weights (excluding negative cycles) where executing Dijkstra's algorithm from a specified vertex s fails to accurately determine the shortest path lengths from s to every node in the graph.

a) Illustrate the accurate representation of the shortest path and contrast it with the path that Dijkstra's algorithm would identify.

b) Propose a solution that can fix this problem of negative edges.

5 Problem Maximum Flow

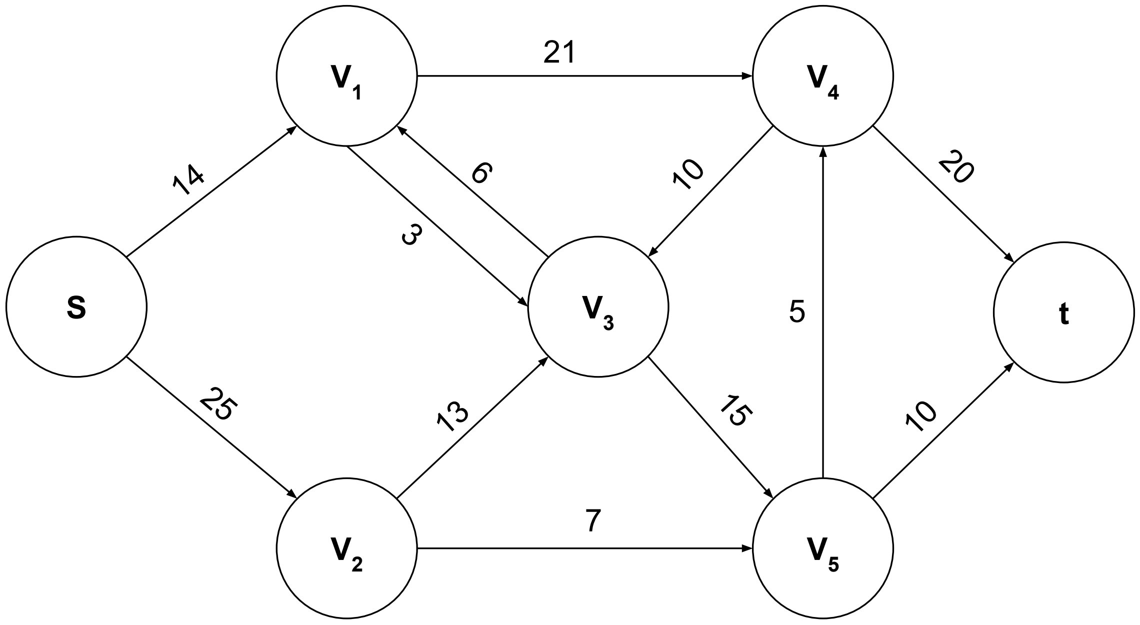
Given the flow network G in Figure 5

a) Resolve the antiparallel edge issue in G.

b) Walk through the Ford-Fulkerson algorithm by hand, starting from source s and ending at sink t, on the flow network G.

c) Show the bottleneck of the flow network by using cuts.

d) Describe the running time of the Ford-Fulkerson algorithm. Suggest any improvement to the Ford-Fulkerson algorithm.


s
V_1
V_2
V_3
V_4
V_5
t
14
25
21
3
6
13
7
10
5
15
20
10