# CSE 208: Data Structures and Algorithms II
## Offline Assignment: Flow & Bipartite Matching

**Department of Computer Science and Engineering**
**Term: July 2025**

---

## Problem 1

In this problem, you are required to implement an algorithm to compute the maximum flow in a directed graph using the **Ford–Fulkerson method.**

You are given a directed graph G=(V, E), where each edge (u,v) $\in$ E has a non-negative integer capacity c(u,v). Two special vertices are designated as the source ***s*** and the sink ***t***. The objective is to determine the maximum amount of flow that can be sent from ***s*** to ***t*** while respecting capacity constraints and flow conservation at intermediate vertices.

### Task Requirements

1. You will implement the **Edmonds–Karp algorithm**, which uses breadth-first search to find augmenting paths and runs in $O(VE^2)$.

2. The program should output:
    - The maximum flow value
    - The final flow on each edge

### Input Format

- The first line of input will have two space-separated non-negative integers N and M, the number of nodes and edges in the graph.
- In the next M lines, there will be three space-separated integers, u, v, c denoting a directed edge (u, v) and its capacity c=c(u,v); ($0 \leq u, v < N$)

- The last line will contain two space-separated non-negative integers, s and t denoting a source and a sink respectively.

## Output Format

- Print the maximum flow from s to t found by your algorithm in the given graph in the first line.
- In the following M lines, you need to print the flow through each edge where each line will contain four integers in the format u v f/c denoting the flow f through the edge (u, v) with capacity c.

## Sample I/O

| Input | Output |
|---|---|
| 6 10<br>0 1 16<br>0 2 13<br>1 2 10<br>1 3 12<br>2 1 4<br>2 4 14<br>3 2 9<br>3 5 20<br>4 3 7<br>4 5 4<br>0 5 | 23<br>0 1 12/16<br>0 2 11/13<br>1 2 0/10<br>1 3 12/12<br>2 1 0/4<br>2 4 11/14<br>3 2 0/9<br>3 5 19/20<br>4 3 7/7<br>4 5 4/4 |

## Problem 2

The Election Commission has two distinct lists of staff who need to be paired up for duty at polling stations.

- Presiding Officers (Group A)
- Polling Agents (Group B)

**The Rule:** Every polling station must have exactly one Presiding Officer and one Polling Agent. You are given a compatibility list showing which Officer can work with which Agent (based on location or language). Your goal is to form the maximum number of compatible pairs.

### Simplifications:

- You do not need to check for validity. **The input is guaranteed to be bipartite.**
- **Nodes 0 to K-1** are always Presiding Officers.
- **Nodes K to N-1** are always Polling Agents.
- Connections only exist between an Officer and an Agent.

### Input Format

- **Line 1:** Three space-separated integers: N, K, M.
    - N: Total number of staff.
    - K: Number of Presiding Officers (IDs 0 to K-1).
    - M: Number of compatibility entries.
- **Next M lines:** Two integers u, v.
    - This denotes that Officer u is compatible with Agent v.

### Output Format

- **Line 1:** The maximum number of teams (pairs) you can form.
- **Following lines:** The specific pairs (u, v) for each team.

**You have to solve this problem by reusing the maximum flow algorithm implemented in Problem 1.**

## Sample I/O

## Input:

6 3 4
0 3
0 4
1 4
2 5

(Explanation: Total 6 people. 3 Officers (0, 1, 2). 3 Agents (3, 4, 5). Officer 0 can work with Agents 3 or 4)

## Output:

3
0 3
1 4
2 5

## Submission Guidelines

- Create a folder named with your 7-digit Student ID.
- Include your source files (e.g., .cpp, .java, or .py)
- Zip the folder into a single .zip file.
- Plagiarism will result in a 100% mark deduction. Ensure the work is your own.