Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science & Statistics

CSU22012

Lab 6
Flow

Head Teaching Assistant:	Ruth Holmes
Teaching Assistant:	Matías Duran, Caoilfhionn Ní Dheoráin
Coordinator:	Anthony Ventresque
Date:	Monday 24th March, 2025
Total Number of Pages:	1

This project explores how we can represent a network of patents and use graph algorithms to analyse their relationships (should remind you of the programming test). By treating each patent as a vertex in a directed graph (where edges indicate citations from one patent to another), we can run algorithms such as BFS/DFS, sorting of patents, and even max flow using Ford–Fulkerson (or Edmond–Karp). The overall goal is for students to see how fundamental data structures and algorithms (like adjacency lists, BFS, sorting, indexing, and max-flow) fit together in a real-world-style scenario: patent citations.

Questions (ask the demonstrators if you are unsure):

- What is a 'bottleneck' in the context of an augmenting path?
- Under what conditions does the algorithm terminate?
- Why might the max flow be zero in some graphs?
- How does the BFS (aka., Edmond–Karp) differ from a standard BFS in an unweighted digraph?
- Why do we add 1 to rand.nextInt(10) when generating random capacities? What happens if we do not?
- Which piece of code specifically updates the flow along the chosen augmenting path? How does it handle forward versus reverse edges?
- How would you add logic to detect if any edges form a cycle that might hold extra capacity? Is that relevant for the max flow solution?
- In the code, where and how do we store the final flow values for each edge, and how can we print them to confirm the algorithm's results?
- How might you modify the code if you wanted integer flow only, or if you wanted double, precision flows?
- How could you alter the code to handle multiple sources or multiple sinks?
- What is the time complexity of Ford–Fulkerson with BFS in the worst case?