Data Structure	Lab 11
Spring 2021	Tuesday, June 29

Lab Goals:

- Work on Projects - First Evaluation (**Deadline**: 30th June, 2021)
- Working with Dijsktra, BFS and DFS (**Deadline:** 13th July, 2021**)**

High Level Project Rubrics: (10 marks) – Practical Evaluation

	Implementation
10	Live deployment
9	Homepage with front and backend.
7	Code repository created on GitLab + Both members share the code via Git + Both members use commit/push and pull techniques for code sharing **+** Rubric IV requirements.
6	Properly formatted and visually pleasant UI/Template integrated **+** Rubric III requirements.
4	Home Screen with user interface and backend **+** Rubric II requirements**.**
2	Registration and Login Screen Completion + Front End + Backend + Rubric 0 requirements
0	Title not approved yet/Nothing done

Lab Rubrics:

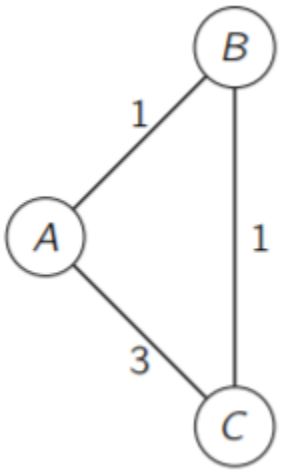
"No plagiarism" text is present in all rubrics.

	Implementation
5	Designed algorithms for Robust Fuel Distribution i.e., solved Q5 + Rubric IV requirements
4	Drew graphs for same vertex order BFS and DFS traversal and different vertex order traversal + Rubric III requirement.
3	Have full understanding of Python notebook code + Run and executed all statements + Understand algorithm working of Dijkstra and BFS codes
1	Completed Pre-lecture exercise

0	Did not attempt any question

Homework Question:

1. Understand and execute Python notebook provided for Dijkstra algorithm. (Attached in folder).
2. **Bonus Optional:** Implement Dijkstra algorithm in C++ with graph class. (optional)
3. **Pre-lecture exercise:** Consider the following graph, where the weights (the numbers on the edges) are meant to represent the cost of walking along an edge



   a. Suppose you do BFS starting at the node A. What does BFS find as the shortest path from A to C?
   b. If you take the weights into account, what is the shortest path from A to C? (By "take the weights into account" we mean add up the weights along each path the get the length of a path. So, with the weights, the cost of the path A → C → B would be 3 + 1 = 4.)
   c. Try to think about how you would design an algorithm to find shortest paths in weighted graphs. Can you modify BFS to do it?

4. **[BFS and DFS Recap.]**

Give one example of a connected undirected graph on four vertices, A, B, C, and D, so that both DFS and BFS search discover the vertices in the **same** order when starting at vertex A. Then give one example of a connected undirected graph on four vertices, A, B, C, and D where BFS and DFS discover the vertices in a **different** order when started at A. Assume that both DFS and BFS iterate over neighbors in alphabetical order.

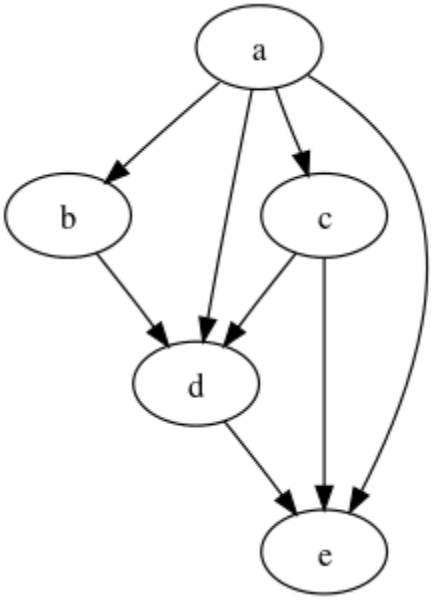
Above, discover means the time that the algorithm first reaches the vertex.

Note on drawing graphs: You might try http://madebyevan.com/fsm/ which allows you to draw graphs with your mouse and convert it into LATEX code. (By default it makes directed graphs; you can add an arrow in both directions to get something that approximates an undirected graph). **Sticky Graph** (http://jzacharyg.github.io/StickyGraph/) is another useful tool for drawing graphs. You can use the +v and +e buttons on the bottom right corner to add nodes and edges, and the TikZ button on the bottom left corner to generate the corresponding LATEX code.

[**We are expecting** A drawing of your two graphs and an ordered list of vertices discovered by BFS and DFS for each of them.]

Material (Wikipedia):

Directed Acyclic Graph: In mathematics, particularly graph theory, and computer science, a directed acyclic graph (DAG or dag) is a directed graph with no directed cycles. That is, it consists of vertices and edges (also called arcs), with each edge directed from one vertex to another, such that following those directions will never form a closed loop.



Cyclic Graph: It is a directed graph that contains a path from at least one node back to itself. In simple terms, cyclic graph contains a cycle.

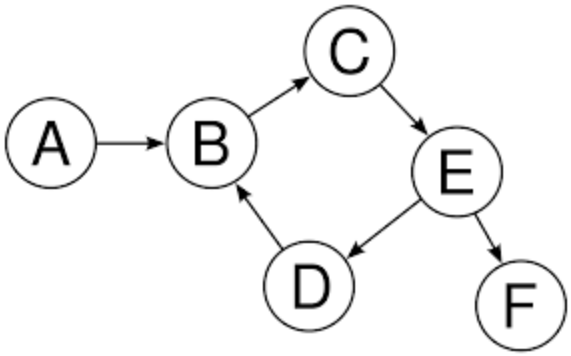


Figure 5 : Cyclic Graph

5. **Robust Fuel Distribution.**

In September 2019, an explosion in Shelby County, Alabama severed the Colonial Pipeline which transports more than 100 million gallons of refined petroleum products from Houston to the southeast every day. The incident caused gasoline prices to jump 15% and might cause jet fuel and gasoline shortages in the region. This event has made you concerned about fuel supply/price stability in the US.

As a student of Data Structures and Algorithms couse, you would like to analyze this system. Here we will consider the robustness of the fuel distribution system under a few simplifications. Let our system be a graph such that:

- Each edge in the graph is a pipe.
- Each edge is directed (fuel flow is not bidirectional).
- Each node in the graph is a combination refinery/distribution terminal (nodes both produce and consume fuel products, so they can have both incoming and outgoing edges).
- Every refinery produces the same fuel that every distribution terminal accepts. In other words, we will not worry about dedicated pipelines for different products.

Assume the system has n nodes and m edges. You would like the system to have stronger robustness by designing algorithms that verify if the system has certain properties:

(a) Design an algorithm that verifies if the fuel distribution system has the following property:

There is a way to transport fuel from any node in the graph to any other node in the graph. Your algorithm should run in O(n + m).

[**We are expecting:** A description of the algorithm, a brief explanation of why the algorithm works and an informal justification of the algorithm's running time.]

(b) Design an algorithm that verifies if the fuel distribution system has the following property:

No single pipe failure can isolate any node from either distributing fuel to the rest of the system or consuming fuel from the rest of the system. That is, removing a single edge cannot cause any node to be unreachable from any other node. Your algorithm should run in O(mn+m^2)). You may assume an algorithm that satisfies the property in part (a) exists and use it directly.

[**We are expecting:** A description of the algorithm, a brief explanation of why the algorithm works and an informal justification of the algorithm's running time.]