UNIVERSIDAD TECNOLÓGICA DE PANAMÁCENTRO REGIONAL DE CHIRIQUÍ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

CARRERA:
Gestión y Desarrollo de Software

ACTIVIDAD No. 10

LABORATORIO No. 5

“Laboratorio 5”

ASIGNATURA: Estructura de Datos II

DOCENTE:
Profa. Nunehar Mondul

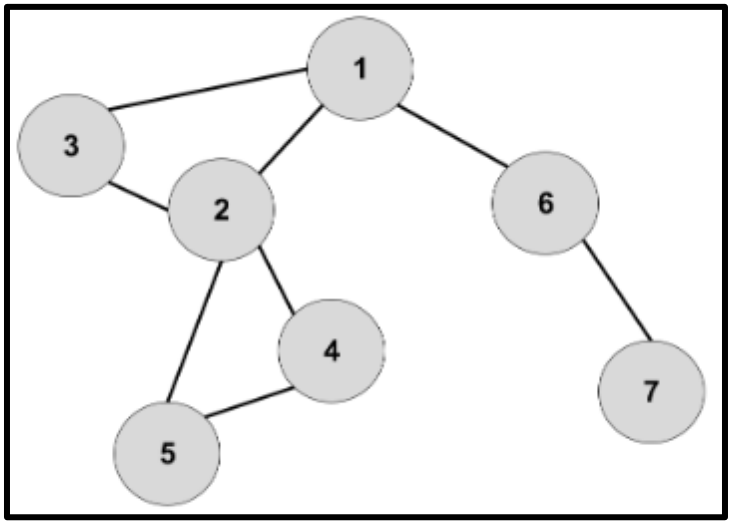
ESTUDIANTE/s:
Jorge Jiménez (4-826-874)
Briant Arango (4-825-620)
-

I SEMESTRE 2025

FECHA:
05/20/2025

Desarrollo

1. "**Mencionar los nodos, las aristas y los caminos del nodo 7 al nodo 5.**" **(BA)**



Nodos: [1, 2, 3, 4, 5, 6, 7]

Aristas: 8

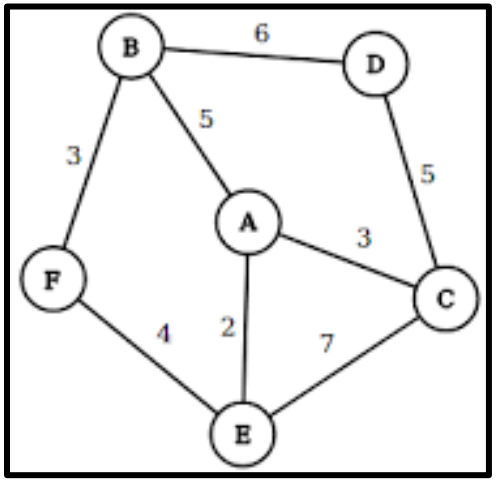
Caminos del nodo 7 al nodo 5: (5, 2) (2,1) (1,6) (6, 7)
(5, 4) (4, 2) (2, 1) (1, 6) (6, 7)
(5, 2) (2, 3) (3, 1) (1, 6) (6, 7)
(5, 4) (4, 2) (2, 3) (3, 1) (1, 6) (6, 7)

2. "Hacer matriz de adyacencia de la 1." **(JJ)**

	1	2	3	4	5	6	7
1	0	1	1	0	0	1	0
2	1	0	1	1	1	0	0
3	1	1	0	0	0	0	0
4	0	1	0	0	1	0	0
5	0	1	0	1	0	0	0
6	1	0	0	0	0	0	1
7	0	0	0	0	0	1	0

3. Con el siguiente grafo, hacer matriz de adyacencia. (BA)

	A	B	C	D	E	F
A	0	0	3	0	2	0
B	5	0	0	6	0	3
C	0	0	0	0	0	0
D	0	0	5	0	0	0
E	0	0	7	0	0	0
F	0	0	0	0	4	0



4. ¿Cuál es la utilidad de los grafos? (BA)

R: Los grafos son utilizados en:

- Redes y comunicación: Los grafos se utilizan para modelar redes de computadoras, sistemas de comunicación y enrutamiento de datos. Los nodos representan dispositivos (como computadoras o routers), y las aristas representan conexiones entre ellos.
- Búsqueda y Recorridos: Algoritmos de búsqueda como Depth-First Search (DFS) y Breadth- First Search (BFS) se aplican en grafos para encontrar caminos, determinar la conectividad y explorar estructuras de datos.
- Grafos Dirigidos Acílicos (DAG): Los DAGs se utilizan en sistemas de planificación y ejecución de tareas, donde las dependencias temporales entre actividades se modelan de manera eficiente.
- Algoritmos de busqueda de caminos: Los grafos se utilizan en aplicaciones de mapas para resolver problemas sobre búsqueda de caminos con el menor costo, por ejemplo, la ruta que usará el taxi para llevar a una persona a su destino.

5. ¿Qué representan los pesos en los grafos, en campos como la logística, computación, geografía? (BA)

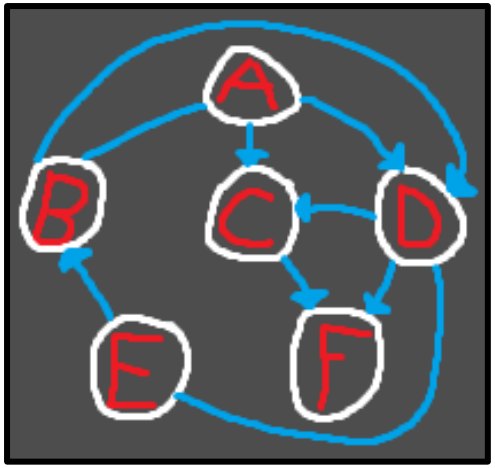
R: En geografía, el peso representa distancia, en computación, la presencia de pesos en las aristas influye en la elección de algoritmos, en logística, modelan situaciones del mundo real donde hay costos asociados a las conexiones, por ejemplo, redes de transporte, rutas de vuelo, redes de telecomunicaciones y planificación de proyectos.

6. Con la siguiente matriz de adyacencia dibujar el grafo resultante, indicar si es dirigido o no. (JJ)

El grafo es ***dirigido*** porque unos nodos conectan con otros que no se conectan devuelta.

(Ej: A conecta con C, pero C no conecta devuelta con A)

$$M = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$



7. Para el siguiente código de grafo documentar y cambiar la lista de adyacencia actual. (JJ)

Grafo.java

```java
package grafos;
import java.util.LinkedList;

public class Grafo {
    private int V; // Número de vértices
    private LinkedList<Integer>[] adj; // Arreglo de listas de adyacencia

    // Constructor del grafo
    public Grafo(int v) {
        V = v;
        adj = new LinkedList[v]; // Inicializa el arreglo
        for (int i = 0; i < v; ++i) {
            adj[i] = new LinkedList<>(); // Crea una lista vacía para cada vértice
        }
    }

    // Método para agregar una arista del nodo v al nodo w
    public void agregarArista(int v, int w) {
        adj[v].add(w); // Agrega w a la lista de adyacencia de v
    }

    // Método para imprimir el grafo
    public void imprimirGrafo() {
        for (int i = 0; i < V; ++i) {
            System.out.println("Lista de adyacencia del nodo " + i + ":");
            for (Integer j : adj[i]) {
                System.out.print(" -> " + j);
            }
            System.out.println();
        }
    }

    // Método principal para ejecutar el código
    public static void main(String[] args) {
        Grafo grafo = new Grafo(4); // Crear un grafo con 4 nodos (0 a 3)

        // Nuevas aristas
        grafo.agregarArista(0, 3);
        grafo.agregarArista(1, 0);
        grafo.agregarArista(1, 2);
        grafo.agregarArista(2, 3);
        grafo.agregarArista(3, 1);

        grafo.imprimirGrafo();
    }
}
```

Output - grafos (run)

```
run:
Lista de adyacencia del nodo 0:
 -> 3
Lista de adyacencia del nodo 1:
 -> 0 -> 2
Lista de adyacencia del nodo 2:
 -> 3
Lista de adyacencia del nodo 3:
 -> 1
BUILD SUCCESSFUL (total time: 0 seconds)
```