UNIVERSIDAD TECNOLÓGICA DE PANAMÁ CENTRO REGIONAL DE CHIRIQUÍ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

CARRERA:
Gestión y Desarrollo de Software

ACTIVIDAD No. 11

LABORATORIO No. 6

"Laboratorio 6"

ASIGNATURA: Estructura de Datos II

DOCENTE:
Profa. Nunehar Mondul

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

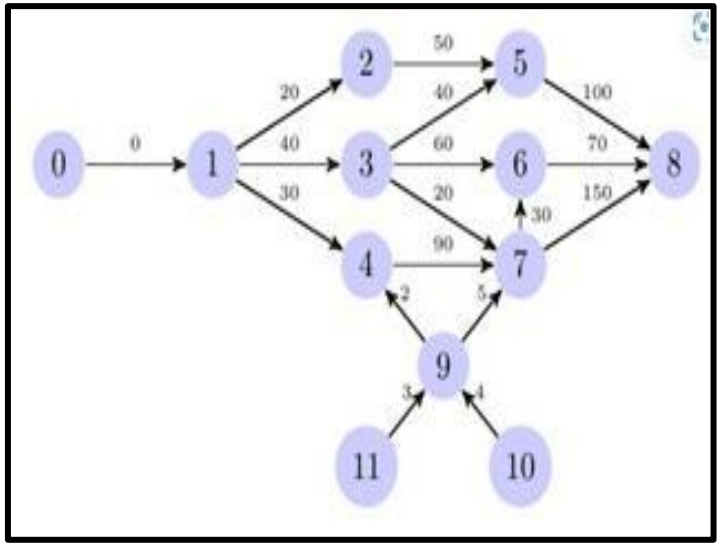
-

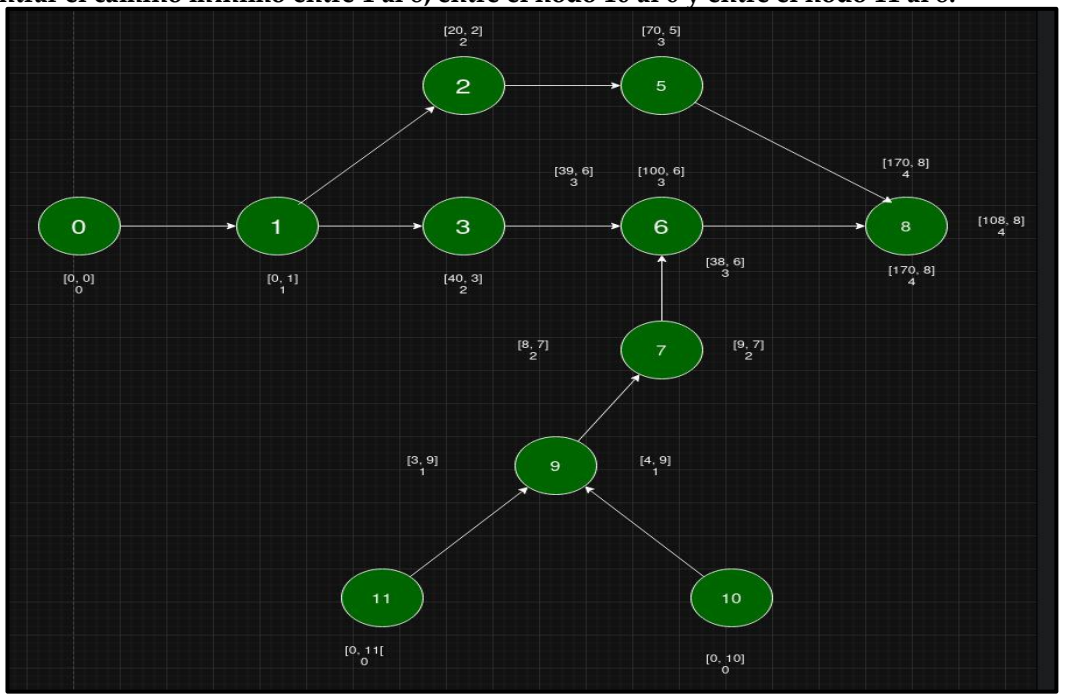
I SEMESTRE 2025

FECHA:
05/27/2025

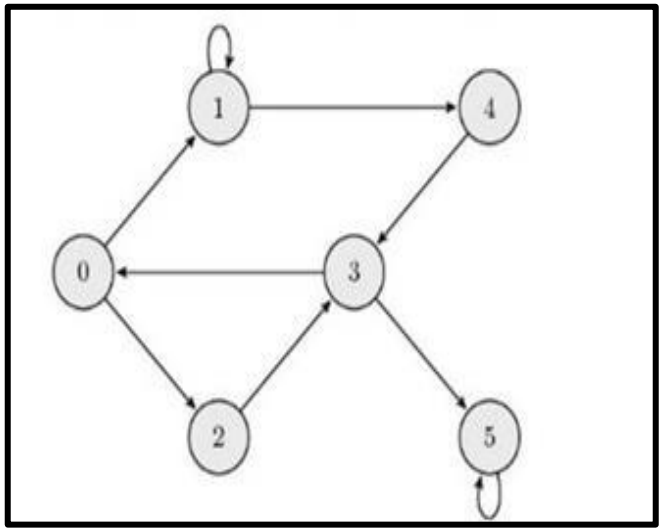
Desarrollo

1. **"Para el siguiente grafo, aplicar algoritmo de Dijkstray encontrar el camino mínimo entre 1 al 8, entre el nodo 10 al 6 y entre el nodo 11 al 8."**





2. **"Al siguiente grafo aplicarle algoritmo de Warshall."**



W0	0	1	2	3	4	5
0	0	1	1	0	0	0
1	0	1	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	0	0	1
4	0	0	0	1	0	0
5	0	0	0	0	0	1

W1	0	1	2	3	4	5
0	0	1	1	0	0	0
1	0	1	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	0	0	1
4	0	0	0	1	0	0
5	0	0	0	0	0	1

W2	0	1	2	3	4	5
0	0	1	1	0	1	0
1	0	1	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	0	0	1
4	0	0	0	1	0	0
5	0	0	0	0	0	1

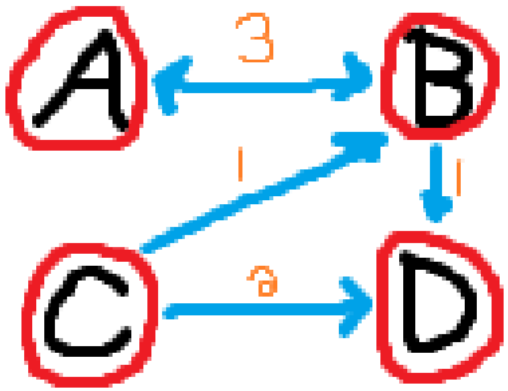
W3	0	1	2	3	4	5
0	0	1	1	1	1	0
1	0	1	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	0	0	1
4	0	0	0	1	0	0
5	0	0	0	0	0	1

W4	0	1	2	3	4	5
0	0	1	1	1	1	1
1	0	1	0	0	1	0
2	0	0	0	1	0	1
3	0	0	0	0	0	1
4	0	0	0	1	0	1
5	0	0	0	0	0	1

W5	0	1	2	3	4	5
0	0	1	1	1	1	1
1	0	1	0	1	1	1
2	0	0	0	1	0	1
3	0	0	0	0	0	1
4	0	0	0	1	0	1
5	0	0	0	0	0	1

3. **"Con la siguiente matriz de adyacencia dibujar grafo y luego aplicar Warshall."**

	A	B	C	D
A	0	3	∞	∞
B	3	0	∞	1
C	∞	1	0	2
D	∞	∞	∞	0



W0	A	B	C	D
A	0	3	∞	∞
B	3	0	∞	1
C	∞	1	0	2
D	∞	∞	∞	0

W1	A	B	C	D
A	0	3	∞	4
B	3	0	∞	1
C	4	1	0	2
D	∞	∞	∞	0

W2	A	B	C	D
A	0	3	∞	4
B	3	0	∞	1
C	4	1	0	2
D	∞	∞	∞	0

W3	A	B	C	D
A	0	3	∞	4
B	3	0	∞	1
C	4	1	0	2
D	∞	∞	∞	0

4. **"Explicar que obtenemos con la última matriz luego de aplicar el algoritmo de Warshall. Analizarlo con el grafo de la 3."**

Con la última matriz luego de aplicar el algoritmo de Warshall, obtenemos la suma de los pesos de los caminos de los grafos. A pesar de que esta suma de pesos se puede hacer a simple vista, la matriz es una confirmación de los valores más que nada.

5. **“Para el siguiente código documentar y luego cambiar grafo para cambiar resultado. Dibujar grafo.”**

FloydWarshall.java

```java
package grafos2;
import java.util.Scanner;

public class FloydWarshall {
    private int[][] DistanceMatrix; // Matriz para almacenar distancias mínimas
    private int numberofvertices; // Cantidad de nodos
    public static final int INFINITY = 999; // Valor para representar "infinito"

    // Constructor que inicializa la matriz de distancias
    public FloydWarshall(int numberofvertices) {
        DistanceMatrix = new int[numberofvertices + 1][numberofvertices + 1];
        this.numberofvertices = numberofvertices;
    }

    // Método que implementa el algoritmo de Floyd-Warshall
    public void floydwarshall(int[][] AdjacencyMatrix) {
        // Copia la matriz original a la matriz de distancias
        for (int source = 1; source <= numberofvertices; source++) {
            for (int destination = 1; destination <= numberofvertices; destination++) {
                DistanceMatrix[source][destination] = AdjacencyMatrix[source][destination];
            }
        }

        // Algoritmo Floyd-Warshall
        for (int intermediate = 1; intermediate <= numberofvertices; intermediate++) {
            for (int source = 1; source <= numberofvertices; source++) {
                for (int destination = 1; destination <= numberofvertices; destination++) {
                    // Verifica si se puede mejorar la distancia a través de un nodo intermedio
                    if (DistanceMatrix[source][intermediate] + DistanceMatrix[intermediate][destination] < DistanceMatrix[source][destination]) {
                        DistanceMatrix[source][destination] = DistanceMatrix[source][intermediate] + DistanceMatrix[intermediate][destination];
                    }
                }
            }
        }

        // Imprime la matriz
        for (int source = 1; source <= numberofvertices; source++) {
            System.out.print("\t" + source);
        }
        System.out.println();
        for (int source = 1; source <= numberofvertices; source++) {
            System.out.print(source + "\t");
            for (int destination = 1; destination <= numberofvertices; destination++) {
                System.out.print(DistanceMatrix[source][destination] + "\t");
            }
            System.out.println();
        }
    }

    public static void main(String... arg) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the number of vertices");
        int numberOfVertices = scan.nextInt();
        int[][] adjacencyMatrix = new int[numberOfVertices + 1][numberOfVertices + 1];

        System.out.println("Enter the Weighted Matrix for the graph");
        for (int source = 1; source <= numberOfVertices; source++) {
            for (int destination = 1; destination <= numberOfVertices; destination++) {
                adjacencyMatrix[source][destination] = scan.nextInt();

                // Costo cero se considera infinito, excepto para bucles
                if (source == destination) {
                    adjacencyMatrix[source][destination] = 0;
                } else if (adjacencyMatrix[source][destination] == 0) {
                    adjacencyMatrix[source][destination] = INFINITY;
                }
            }
        }

        System.out.println("The Transitive Closure of the Graph");
        FloydWarshall floydwarshall = new FloydWarshall(numberOfVertices);
        floydwarshall.floydwarshall(adjacencyMatrix);
        scan.close();
    }
}
```

```
Enter the number of vertices
4
Enter the Weighted Matrix for the graph
0 3 999 999
3 0 999 1
999 1 0 2
999 999 999 0
The Transitive Closure of the Graph
        1       2       3       4
1       0       3       999     4
2       3       0       999     1
3       4       1       0       2
4       999     999     999     0
```