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

CARRERA:
Gestión y Desarrollo de Software

ACTIVIDAD No. 12

LABORATORIO No. 7

"Laboratorio 7"

ASIGNATURA: Estructura de Datos II

DOCENTE:
Profa. Nunehar Mondul

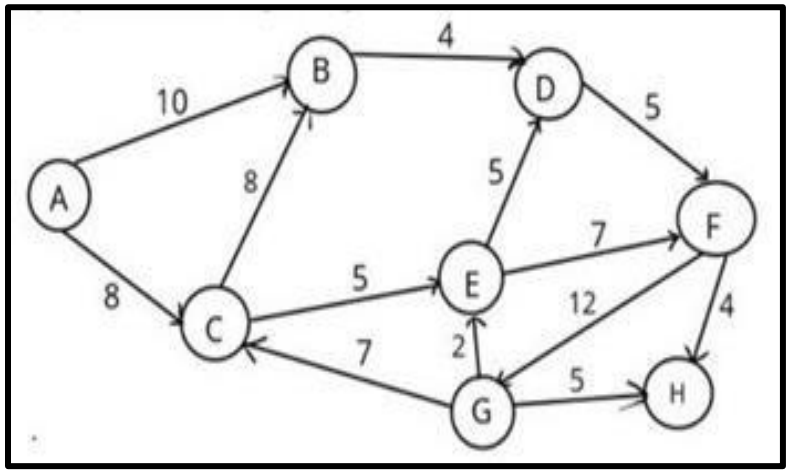
ESTUDIANTE/s:
Jorge Jiménez (4-826-874)
-
-

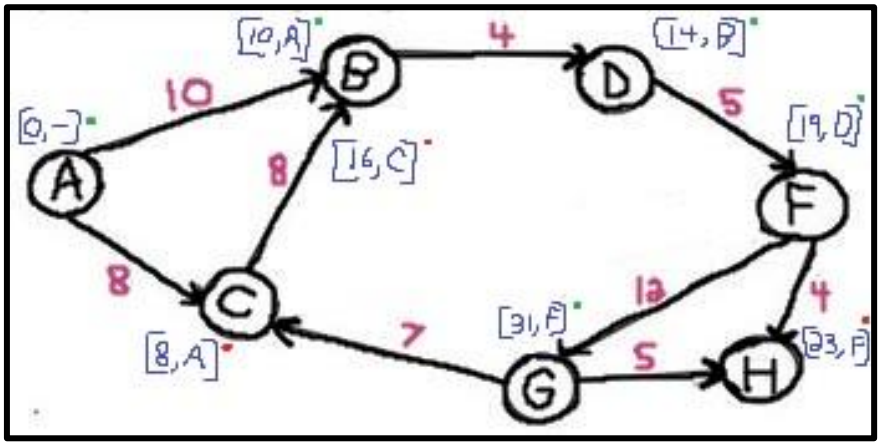
I SEMESTRE 2025

FECHA:
06/03/2025

Desarrollo

1. **"Para el siguiente grafo, eliminar nodo E. Y luego correr Dijktra de A a G. Analizar que nodo es fundamental para llegar de A a G y que no se puede borrar. Hacer matriz de adyacencia."**

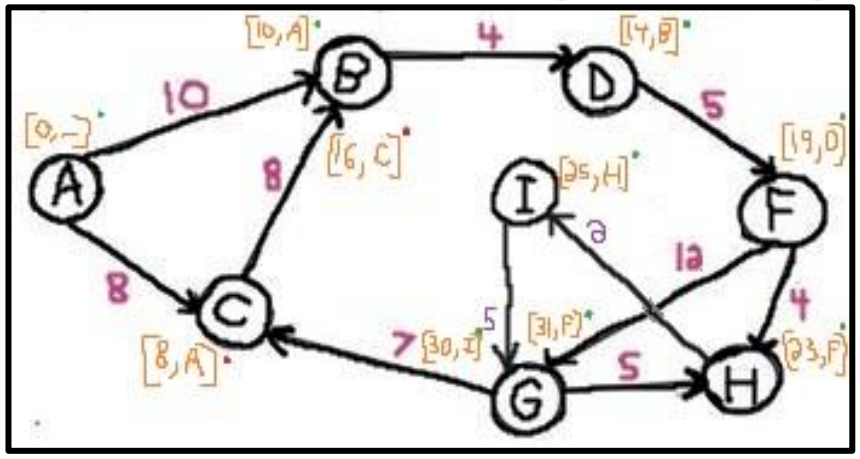




Nodo fundamental: la **B**, porque es el único nodo que conecta con lo demás a la derecha al eliminar la E.

	A	B	C	D	F	G	H
A	0	10	8	∞	∞	∞	∞
B	∞	0	∞	4	∞	∞	∞
C	∞	8	0	∞	∞	∞	∞
D	∞	∞	∞	0	5	∞	∞
F	∞	∞	∞	∞	0	12	4
G	∞	∞	7	∞	∞	0	5
H	∞	∞	∞	∞	∞	∞	0

2. **Añadir nodo I, conectar arista de H a I con peso 2 y de I a G con peso 5. Correr Dijktra con el nuevo nodo. Dibujar nueva matriz de adyacencia.**



	A	B	C	D	F	G	H	I
A	0	10	8	∞	∞	∞	∞	∞
B	∞	0	∞	4	∞	∞	∞	∞
C	∞	8	0	∞	∞	∞	∞	∞
D	∞	∞	∞	0	5	∞	∞	∞
F	∞	∞	∞	∞	0	12	4	∞
G	∞	∞	7	∞	∞	0	5	∞
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	∞	∞	∞	∞	5	∞	0

3. Con la matriz de adyacencia del 2 correr algoritmo de Warshall.

W0	A	B	C	D	F	G	H	I
A	0	10	8	∞	∞	∞	∞	∞
B	∞	0	∞	4	∞	∞	∞	∞
C	∞	8	0	∞	∞	∞	∞	∞
D	∞	∞	∞	0	5	∞	∞	∞
F	∞	∞	∞	∞	0	12	4	∞
G	∞	∞	7	∞	∞	0	5	∞
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	∞	∞	∞	∞	5	∞	0

W1	A	B	C	D	F	G	H	I
A	0	10	8	14	∞	∞	∞	∞
B	∞	0	∞	4	∞	∞	∞	∞
C	∞	8	0	12	∞	∞	∞	∞
D	∞	∞	∞	0	5	∞	∞	∞
F	∞	∞	∞	∞	0	12	4	∞
G	∞	∞	7	∞	∞	0	5	∞
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	∞	∞	∞	∞	5	∞	0

W2	A	B	C	D	F	G	H	I
A	0	10	8	14	∞	∞	∞	∞
B	∞	0	∞	4	∞	∞	∞	∞
C	∞	8	0	12	∞	∞	∞	∞
D	∞	∞	∞	0	5	∞	∞	∞
F	∞	∞	∞	∞	0	12	4	∞
G	∞	∞	7	∞	∞	0	5	∞
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	∞	∞	∞	∞	5	∞	0

W3	A	B	C	D	F	G	H	I
A	0	10	8	14	∞	∞	∞	∞
B	∞	0	∞	4	∞	∞	∞	∞
C	∞	8	0	12	∞	∞	∞	∞
D	∞	∞	∞	0	5	∞	∞	∞
F	∞	∞	∞	∞	0	12	4	∞
G	∞	15	7	19	∞	0	5	∞
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	∞	∞	∞	∞	5	∞	0

W4	A	B	C	D	F	G	H	I
A	0	10	8	14	19	∞	∞	∞
B	∞	0	∞	4	9	∞	∞	∞
C	∞	8	0	12	17	∞	∞	∞
D	∞	∞	∞	0	5	∞	∞	∞
F	∞	∞	∞	∞	0	12	4	∞
G	∞	15	7	19	24	0	5	∞
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	∞	∞	∞	∞	5	∞	0

W5	A	B	C	D	F	G	H	I
A	0	10	8	14	19	31	23	∞
B	∞	0	∞	4	9	21	13	∞
C	∞	8	0	12	17	29	21	∞
D	∞	∞	∞	0	5	17	9	∞
F	∞	∞	∞	∞	0	12	4	∞
G	∞	15	7	19	24	0	5	∞
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	∞	∞	∞	∞	5	∞	0

W6	A	B	C	D	F	G	H	I
A	0	10	8	14	19	31	23	∞
B	∞	0	28	4	9	21	13	∞
C	∞	8	0	12	17	29	21	∞
D	∞	32	24	0	5	17	9	∞
F	∞	27	19	31	0	12	4	∞
G	∞	15	7	19	24	0	5	∞
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	20	12	24	29	5	10	0

W7	A	B	C	D	F	G	H	I
A	0	10	8	14	19	31	23	25
B	∞	0	28	4	9	21	13	15
C	∞	8	0	12	17	29	21	23
D	∞	32	24	0	5	17	9	11
F	∞	27	19	31	0	12	4	6
G	∞	15	7	19	24	0	5	7
H	∞	∞	∞	∞	∞	∞	0	2
I	∞	20	12	24	29	5	10	12

W7	A	B	C	D	F	G	H	I
A	0	10	8	14	19	31	23	25
B	∞	0	28	4	9	21	13	15
C	∞	8	0	12	17	29	21	23
D	∞	32	24	0	5	17	9	11
F	∞	27	19	31	0	12	4	6
G	∞	15	7	19	24	0	5	7
H	∞	22	14	26	31	7	0	2
I	∞	20	12	24	29	5	10	12

4. Que hace a un nodo fundamental para el funcionamiento del grafo. Qué pasa cuando no se puede recorrer el grafo por quedar incomunicado.

Un nodo fundamental es aquel nodo cuya eliminación *rompe* la conectividad de una parte importante del grafo. Además, si el grafo queda incomunicado, ya no es posible recorrerlo completamente ni aplicar correctamente algoritmos de caminos o cierre.

5. **Para el siguiente código documentar y luego cambiar los nodos que se insertan y solo eliminar 1.**

Grafos3.java

Source | History

```java
package grafos3;
public class Grafos3 {

  // Clase interna que representa un nodo de la lista
  class DemoNode {
    int NodeData;            // Valor del nodo
    DemoNode NextNode;       // Referencia al siguiente nodo

    public DemoNode(int NodeData) {
      this.NodeData = NodeData;
      this.NextNode = null;
    }
  }

  // Referencias al primer y último nodo de la lista
  public DemoNode HeadNode = null;
  public DemoNode TailNode = null;

  // Método para añadir un nodo al final de la lista
  public void Add_Node(int NodeData) {
    DemoNode NewNode = new DemoNode(NodeData);
    if (HeadNode == null) {              // Si la lista está vacía
      HeadNode = NewNode;
      TailNode = NewNode;
    } else {                             // Si no está vacía, agregar al final
      TailNode.NextNode = NewNode;
      TailNode = NewNode;
    }
  }

  // Método para eliminar el último nodo de la lista
  public void DeleteNode() {
    if (HeadNode == null) {
      System.out.println("List is empty");
      return;
    } else {
      if (HeadNode != TailNode) {
        DemoNode current = HeadNode;
        // Avanzar hasta el penúltimo nodo
        while (current.NextNode != TailNode) {
          current = current.NextNode;
        }
        TailNode = current;          // El nuevo último nodo
        TailNode.NextNode = null;
      } else {                       // Si solo hay un nodo
        HeadNode = TailNode = null;
      }
    }
  }

  // Método para mostrar la lista
  public void DisplayNode() {
    DemoNode CurrentNode = HeadNode;
    if (HeadNode == null) {
      System.out.println("List is empty");
      return;
    }
    while (CurrentNode != null) {
      System.out.print(CurrentNode.NodeData + " ");
      CurrentNode = CurrentNode.NextNode;
    }
    System.out.println();
  }

  // Método principal
  public static void main(String[] args) {
    Grafos3 Linked_List = new Grafos3();

    // Cambiar nodos insertados aquí
    Linked_List.Add_Node(20);
    Linked_List.Add_Node(11);
    Linked_List.Add_Node(22);
    Linked_List.Add_Node(50);

    System.out.println("The Original List is: ");
    Linked_List.DisplayNode();

    // Eliminar solo un nodo (el último)
    Linked_List.DeleteNode();
    System.out.println("The list after deleting the last node: ");
    Linked_List.DisplayNode();
```

Output - grafos3 (run)

```
run:
The Original List is:
20 11 22 50
The list after deleting the last node:
20 11 22
BUILD SUCCESSFUL (total time: 0 seconds)
```