Ejercicios Propios - Teoría de Grafos

Ejercicio 1 – Grafo de Precedencia

Explicación:

En este ejercicio desarrollé un ejemplo propio donde se simula un sistema de monitoreo climático. El propósito es representar las dependencias entre procesos que ocurren en una secuencia lógica. Esta representación se hace mediante un grafo dirigido (de precedencia) que define qué proceso debe ejecutarse antes que otro.

Procesos definidos:

• S1: Sensor de temperatura

• S2: Sensor de humedad

• G: Guardar datos

• P: Procesar los datos

• A: Generar una alerta si se detectan valores anormales

• R: Repetir el ciclo de sensado

Relaciones (dependencias):

• Para guardar datos (G) se deben obtener previamente desde ambos sensores (S1 y S2).

• Para procesar los datos (P), primero deben estar guardados.

• Para generar una alerta (A), debe haberse realizado el procesamiento.

• Finalmente, se repite el sensado (R) solo si ya se alertó o se validó el estado.

¿Qué hace el programa?

El programa fue implementado usando HTML, CSS y JavaScript con la librería D3.js y permite:

• Visualizar el grafo de precedencia directamente en un navegador.

• Ejecutar paso a paso los procesos involucrados haciendo clic en un botón.

• Mostrar la ruta crítica como un recorrido resaltado independiente.

Y mostrar el costo total de la ruta critica

• y asi Entender fácilmente el orden de ejecución y la lógica del flujo de procesos, imitando cómo lo haría un sistema operativo o un sistema de control industrial.

Ejercicio 2 – Árbol Binario de Búsqueda

Explicación:

Este ejercicio consiste en la implementación de un árbol binario de búsqueda que permite localizar un número dado de forma eficiente. La estructura del árbol sigue la lógica de que cada nodo tiene un subárbol izquierdo con valores menores y un subárbol derecho con valores mayores.

Estructura del árbol:

```
    15
   /  \
  7    22
 /\    /\
3 10  18 27
```

Lógica del algoritmo:

1. Se compara el número buscado con la raíz.

2. Si es igual, se ha encontrado.

3. Si es menor, se repite el proceso en el subárbol izquierdo.

4. Si es mayor, se repite en el subárbol derecho.

5. El proceso continúa recursivamente hasta encontrar el número o llegar a una rama vacía (null).

¿Qué hace el programa?

La solución está implementada en HTML, CSS y JavaScript con D3.js, y permite:

• Visualizar el árbol binario completo en pantalla.

• Ingresar un valor y simular paso a paso su búsqueda.

• Mostrar visualmente en el árbol el recorrido de búsqueda.

• Indicar si el valor fue encontrado o no, resaltando los nodos visitados.

y asi Esta herramienta facilita la comprensión del funcionamiento interno de los árboles de decisión binaria y cómo optimizan la búsqueda de datos.