💻 Trabajo Práctico Cuatrimestral – Programación III

Objetivo

Aplicar las tres técnicas principales vistas en el curso (**Divide & Conquer, Greedy y Programación Dinámica**) a problemas concretos. El trabajo debe incluir:

- Pseudocódigo detallado de cada solución.
- Análisis de complejidad temporal (mejor, peor y promedio si aplica).
- Implementación en un **lenguaje de programación elegido por el grupo** (Python, Java, C#, VB.NET, etc.).
- Comparación y conclusiones finales.

Problemas a Resolver

🔹 Parte 1 – Divide & Conquer

Problema: Par de puntos más cercanos

En un plano bidimensional se tienen **n puntos con coordenadas (x, y)**. El objetivo es encontrar el par de puntos que estén a la menor distancia posible entre sí.

1. **Solución ingenua ($O(n^2)$)**
2. **Solución con Divide & Conquer**

👉 Se espera:

- Pseudocódigo del algoritmo.
- Explicación paso a paso.
- Comparación de complejidad con la solución $O(n^2)$.
- Implementación en código real con al menos 5 ejemplos de prueba.

🔹 Parte 2 – Greedy

Problema: Selección de actividades

Un estudiante tiene **n actividades**, cada una con un horario de inicio y fin. El objetivo es seleccionar la **máxima cantidad de actividades que no se superpongan**.

👉 Se espera:

- Pseudocódigo Greedy.
- Justificación de por qué funciona
- Complejidad
- Implementación en código real con casos de prueba.

🔹 Parte 3 – Programación Dinámica

Problema: Mochila 0/1 (Knapsack Problem)

Un ladrón tiene una mochila con capacidad de peso **W**. Hay **n objetos**, cada uno con un **peso w_i** y un **valor v_i**. El objetivo es seleccionar un subconjunto de objetos que **maximice el valor total sin superar la capacidad de la mochila**.

Ejemplo:

- Mochila con capacidad W=10.
- Objetos:
  - O1: peso 6, valor 30
  - O2: peso 3, valor 14
  - O3: peso 4, valor 16
- Solución óptima: {O2, O3} con valor = 30.

👉 Se espera:

- Pseudocódigo del algoritmo.
- Complejidad
- Implementación en el lenguaje elegido.
- Pruebas con diferentes capacidades y objetos.

🔹 Parte 4 – Grafos: Floyd-Warshall

Consigna:
Cada grupo deberá **inventar un problema de la vida real** que pueda modelarse con un

grafo ponderado, y resolverlo utilizando el **algoritmo de Floyd-Warshall** para obtener **las distancias más cortas entre todos los pares de nodos**.

👉 Se espera:

1. **Modelado del problema**:
    - Descripción inventada del caso real.
    - Justificación de por qué se puede modelar como un grafo.
    - Representación del grafo (matriz de adyacencia).
2. **Aplicación del algoritmo**:
    - Pseudocódigo del algoritmo de Floyd-Warshall.
    - Ejecución paso a paso sobre un ejemplo pequeño (4-5 nodos).
    - Complejidad temporal
3. **Implementación en código real**:
    - Lenguaje elegido por el grupo.
    - Mostrar la tabla final de distancias mínimas.
4. **Discusión**:
    - Explicar qué significan los resultados en el contexto inventado.
    - Comparar con usar Dijkstra para un solo nodo (opcional).

Entregables

1. **Informe escrito**
    - Problema explicado con ejemplos.
    - Pseudocódigo de cada solución.
    - Análisis de complejidad (mejor/peor caso).
    - Comparación entre paradigmas.
2. **Código fuente**
    - Implementación en un lenguaje elegido.
    - Comentarios y ejemplos de ejecución.
3. **Conclusiones**
    - Reflexión: cuál técnica fue más sencilla/difícil.
    - Comparación de tiempos reales de ejecución