Taller 3
Análisis y diseño de algoritmos

Carlos Andres Delgado S, Msc
carlos.andres.delgado@correounivalle.edu.co
Noviembre de 2023

Aporte: En el curso de ADA I este taller aporta al RA2.

1. **(35 %) Análisis algoritmo Divide y vencerás.**

   El siguiente algoritmo de ordenamiento es conocido como `StoogeSort`.

   ```
   STOOGE-SORT(A,i,j)
   1 if A[i] > A[j]
   2     then exchange A[i] ↔ A[j]
   3 if i + 1 ≥ j
   4     then return
   5 k ← ⌊(j − i + 1)/3⌋          ▷ Round down.
   6 STOOGE-SORT(A,i,j − k)       ▷ First two-thirds.
   7 STOOGE-SORT(A,i + k,j)       ▷ Last two-thirds.
   8 STOOGE-SORT(A,i,j − k)       ▷ First two-thirds again.
   ```

   a) (5 %) Explique la estrategia: Dividir, conquistar y combinar

   b) (10 %) Implemente el algoritmo

   c) (5 %) Realice pruebas con arreglos aleatorios de tamaño 10, 100, 1000, 10000.

   d) (15 %) Calcule la complejidad teórica del algoritmo y compárela con la complejidad práctica encontrada. Analice lo que encuentra.

2. **(35 %) Diseño algoritmo Divide y vencerás.**
   Diseñe un algoritmo bajo la estrategia divide y vencerás para encontrar la moda de un vector. La moda de un vector es el elemento que más se repite, si existe más de una moda, el algoritmo retornará todos los elementos que son moda. Ejemplo

   - (1,2,2,3,4) La moda es 2
   - (1,2,2,3,3,5) La moda es 2 y 3

   a) (5 %) Explique la estrategia: Dividir, conquistar y combinar

   b) (10 %) Implemente el algoritmo

c) (5 %) Realice pruebas con arreglos aleatorios de tamaño 10, 100, 1000, 10000.

d) (15 %) Calcule la complejidad teórica del algoritmo y compárela con la complejidad práctica encontrada. Analice lo que encuentra.

3. **(30 %)** Comparación ejecución algoritmos

   Implemente los algoritmos QuickSort, Insertion-Sort y Merge-Sort y realice una comparación entre los utilizando una tabla para entradas aleatorias de tamaño 10, 50, 100, 500, 1000, 2000, 5000 y 10000.

   El análisis debe estar contenido en una tabla donde, para cada algoritmo, se tomen muestras de diferentes tamaños (unas 2 o 3 muestras por cada n, por lo que debe tomar el tiempo promedio), se de el tiempo real de ordenamiento, la complejidad temporal del algoritmo, y se saque el factor constante[1].
   Un ejemplo de la tabla (para un algoritmo de complejidad $\mathcal{O}(n^2)$) es el siguiente:

Entrada (n)	Tiempo Real (seg.)	Complejidad ($\mathcal{O}(n^2)$)	Constantes
5	0.11	25	0.0044
5	0.11	25	0.0044
5	0.12	25	0.0048
10	0.38	100	0.0038
10	0.40	100	0.0040
10	0.41	100	0.0041
⋮			
		Constante:	0.0042

Entrega

1. La fecha máxima de entrega de este Taller es el **Sábado 02 de Diciembre de 2023 a las 23:59:59** en un repositorio de Github cuyo enlace será entregado en el campus virtual, se admiten entregas retrasadas pero estas tienen una penalidad de 0.3 acumulada por hora o fracción de retraso. Debe asegurarse que el docente lo pueda acceder haciéndolo público o agregándolo como colaborador si es privado (usando su correo institucional).

2. Este taller puede ser trabajado en grupos de máximo 3 personas.

3. El repositorio de Github debe tener el informe en formato PDF donde se responden los puntos del taller, la calidad de presentación, calidad de escritura y calidad de análisis serán tenidos en cuenta para la nota de cada punto.

4. Estructurar el repositorio de Github con 3 carpetas cada una que contiene las implementaciones realizadas para cada punto, esto será tenido en cuenta para el calculo de la nota.

5. Incluya un archivo de README.md en cada carpeta explicando cómo hacer funcionar su código, debe ser claro con las instrucciones para que el docente pueda evaluar su código, se tomará como no entregado sino hay instrucciones claras sobre las dependencias y pasos para hacer funcionar su solución, no incluya programas dependientes del sistema operativo.

[1] Recuerde que $Tiempo\ Real = Constante * Complejidad$

6. Incluir una sección de discusión de resultados para cada punto, esto será tenido en cuenta para el calculo de la nota.

7. Haga el informe lo más objetivo posible, no incluya información innecesaria como capturas de código, si requiere explicar algún código puede referirse únicamente a la función o método que considere importante, además no incluya las definiciones de los algoritmos, ya que estas son conocidas, concéntrese en explicar su implementación, análisis y a responder los puntos del taller con argumentaciones claras y concisas.