Taller 1
Fundamentos de análisis y diseño de algoritmos

Carlos Andres Delgado S, Ing [*]

Marzo 2018

1. Reglas del taller

1. El taller debe ser entregado el **Miércoles, 21 de Marzo de 2018 08:00am** hora de Colombia, por el campus virtual. El enlace se cierra a esa hora y no le permitirá enviar después de eso.
2. El informe debe ir en formato PDF. Los formatos editables (docx, odt, etc) tienen problemas a la hora de ver el archivo en diferentes versiones. Si no entra en este formato se sanciona 0.5 en la nota del taller
3. Debe incluir en el informe los nombres completos y códigos de los integrantes del grupo. Si no incluye esto, se sanciona 0.5 nota del taller.
4. El código debe ir comentado explicando brevemente cada función y apartes importantes del código. Se debe indicar para que se utiliza en el algoritmo. Si no incluye esto, se sanciona 0.5 nota del taller.
5. No se permite división de grupos ni cambios de integrantes de última hora. No se recibirá el taller si esto va producir conflictos entre los integrantes.
6. Debe entregar el código fuente organizado en carpetas dentro del primer nivel del archivo comprimido, no cree una jerarquía compleja difícil de revisar. Si no realiza este paso se sanciona 0.2 en la nota. Lo importante es que eso sea claro para la revisión.
7. No se permite copiar código de Internet ni de sus compañeros. Si se encuentra código copiado de alguna parte el taller será anulado por completo.
8. Entregue un sólo archivo comprimido. No entregue archivos comprimidos dentro de archivos comprimidos, ya que esto dificulta la revisión enormemente. Si hace esta mala práctica se aplica una sanción con 0.5 en la nota del taller.
9. El informe debe tener buena presentación, esto hace parte de la nota de los puntos del taller
10. No deje que el enlace del campus se cierre. No se reciben trabajos por correo, no insista. Si el campus falla, esto será verificado con la DINTEV. Recuerde estar autenticado ya que el curso permite acceso a invitados
11. Las primeras líneas de cada archivo de código fuente, deben tener los integrantes del grupo con sus nombres y código completos. Si no cumple esto, será sancionado con 0.5 en la nota del taller.

[*] carlos.andres.delgado@correounivalle.edu.co

2. Problemas

1. **(20 %) Comparación de tiempos de ejecución**
   Suponga un computador con un procesador de 1GHz (Hace 1000 millones de operaciones por segundo) y otro de 100Mhz (Hace 100 millones de operaciones por segundo). Dé el tiempo de ejecución en la unidad de tiempo que considere más adecuada (nanosegundos, microsegundos, segundos, minutos, horas, días, años).

T(n) \ n	10^0	10^1	10^3	10^5	10^{10}	10^{20}	10^{30}
$\log n$							
$\sqrt{n}$							
n							
$n \log n$							
n^2							
n^3							
2^n							
$n!$							

   Debe entregar dos tablas. Haga un análisis comparando los resultados que obtiene. Analice cómo influye la capacidad de la máquina (velocidad procesador) en los tiempos de ejecución.

2. **(35 %)** Implemente los siguientes algoritmos de ordenamiento:

   - Insertion Sort
   - Selection Sort

   Calcule la complejidad teórica analizando el código. Recuerde analizar mejor, caso promedio y peor caso. Para esto construya una tabla para cada algoritmo bajo la siguiente estructura.

Linea	**Código**	**Ejecuciones**
1	for(int i=0; i<n; i++){	n+1
2	a=b+c	n

   Debe realizar ejecuciones con arreglos de tamaño 100,10000, 100000 y 1000000. Encuentre la forma de generarlos. La idea es que pueda evaluar los 3 casos.

   Realice al menos 5 ejecuciones de cada tamaño (en total serían 20 por algoritmo) y tome el tiempo promedio de ejecución.

   En Linux puede usar el comando **time** para tomar el tiempo de ejecución en el sistema (Linea *System*) como evidencia de ejecución, o en su lugar use alguna librería de tiempo dentro del lenguaje para tomar el tiempo que se tarda en ordenar una entrada. Tome el tiempo promedio para cada ejecución, la idea es generar una tabla así:

Algoritmo \ n	100	10000	100000	1000000
Insertion Sort (Mejor caso)	x ms	x ms	...	...
Insertion Sort (Caso promedio)	...	...	...	...
Insertion Sort (Peor caso)	...	...	...	...

   Realice un análisis comparando las complejidad teórica contra la complejidad encontrada. Analice tomando en cuenta cómo crece la entrada y lo que encuentra en las ejecuciones.

3. **(25 %)** Solucionar los puntos del problema *2-3 Correctness of Horner's rule* del Libro (página 41). Adicionalmente, debe realizar la implementación de la solución. Calcular la complejidad teórica como en el anterior punto y contrastarla contra la teórica práctica encontrada. Aplique la misma metodología de acuerdo al anterior punto.

4. **(20 %) Encontrando el entero perdido**
   Un arreglo $A[1..n]$ contiene todos los enteros de 0 a n excepto uno. Sería fácil determinar el entero perdido en un tiempo $O(n)$ usando un arreglo auxiliar $B[1..n]$ para registrar cuales números aparecen en A.
   En este problema, sin embargo, no se puede acceder un entero en A con una sola operación. Los elementos de A son representados en binario, y la única operación que se usa para accederlos es "obtenga el j-ésimo bit de $A[i]$", el cual toma tiempo constante.

   Muestre en forma general que si sólo se usa esta operación, aún se puede determinar el entero perdido en un tiempo $O(n)$. Genere al menos 3 ejemplos que respalden su argumentación.