Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC 2133 – Estructuras de Datos y Algoritmos
Interrogación 1

Hora inicio: 14:00 lunes 3 de mayo
Hora máxima de entrega: 23:59 lunes 3 de mayo en SIDING

Esta prueba consiste en 4 preguntas de materia y una pregunta de formalidad (pregunta 0). Responda las preguntas 0 y 1. Luego, de las preguntas 2, 3 y 4, elija **solo dos** para responder. En total, usted debe responder 3 preguntas de materia (incluyendo, obligatoriamente, la pregunta 1) y una pregunta de formalidad (también obligatoria).

Responda las siguientes preguntas:

0. Formalidad. Responde esta pregunta en papel y lápiz, incluyendo tu firma al final.
    a. ¿Cuál es tu nombre completo?
    b. ¿Te comprometes a no preguntar ni responder dudas de la prueba a nadie que no sea parte del cuerpo docente del curso, ya sea de manera directa o indirecta?

1. [Introducción] A pesar de que ***mergeSort*** tiene un comportamiento $\boldsymbol{O}(\boldsymbol{n} \cdot \log \boldsymbol{n})$ en el peor caso e ***insertionSort*** tiene un comportamiento $\boldsymbol{O}(\boldsymbol{n}^2)$ en el peor caso, ***insertionSort*** en la práctica funciona más rápido **para problemas pequeños**. Por tanto, tiene sentido ahorrar recursiones de ***mergeSort*** usando ***insertionSort*** cuando los subproblemas se hacen suficientemente pequeños.

    [Pregunta] Sea $\boldsymbol{n}$ la cantidad de elementos en una secuencia a ordenar y $\boldsymbol{k}$ un valor a determinar, con $\boldsymbol{k} \leq \boldsymbol{n}$. Considera una modificación de ***mergeSort*** llamada ***mergeInserSort*** en la que $\boldsymbol{n}/\boldsymbol{k}$ sublistas de largo $\boldsymbol{k}$ son ordenadas usando ***insertionSort*** y luego unidas usando el mecanismo de ***merge*** conocido.

    a. Muestra que con ***insertionSort*** se pueden ordenar $\boldsymbol{n}/\boldsymbol{k}$ sublistas (por separado; sin mezclarlas), cada una de largo $\boldsymbol{k}$, obteniendo finalmente $\boldsymbol{n}/\boldsymbol{k}$ sublistas ordenadas, en tiempo $\boldsymbol{O}(\boldsymbol{nk})$ en el peor caso.

    b. Muestra cómo se pueden mezclar las sublistas ordenadas, obteniendo finalmente una sola lista ordenada, en tiempo $\boldsymbol{O}(\boldsymbol{n} \cdot \log(\boldsymbol{n}/\boldsymbol{k}))$ en el peor caso.

    c. Dado que nuestro ***mergeInserSort*** corre en tiempo $\boldsymbol{O}(\boldsymbol{nk} + \boldsymbol{n} \cdot \log(\boldsymbol{n}/\boldsymbol{k}))$ en el peor caso, ¿cuál es el máximo valor de $\boldsymbol{k}$, en función de $\boldsymbol{n}$ (<u>en notación $\boldsymbol{O}$</u>), para el cual ***mergeInserSort*** corre en el mismo tiempo (<u>en notación $\boldsymbol{O}$</u>) que ***mergeSort*** normal? Hint: $\log(\log(\boldsymbol{n}))$ es despreciable, relativo a $\log(\boldsymbol{n})$, para $\boldsymbol{n}$ suficientemente grande.

    d. ¿Qué deberías considerar (aparte del comportamiento $\boldsymbol{O}$) para elegir el valor de $\boldsymbol{k}$ en la práctica? Comente brevemente.

2. A un alumno que estaba estudiando ***quickSort*** se le ocurrió que podría usar el algoritmo ***median*** visto en clases para mejorar ***quickSort***. Simplemente modifica ***median*** para que retorne el índice de la mediana (en vez del valor de la mediana), y reemplaza la llamada a ***partition*** por una llamada a ***median* modificado** en la definición de ***quickSort***. Así, no tendríamos que arriesgarnos a elegir malos pivotes, pues ya sabemos encontrar la mediana y podemos lograr siempre el mejor caso de ***quickSort***. Nota: asume que no hay elementos repetidos.
    a. ¿Cuál es la complejidad de la idea de tu compañero en el mejor caso?
    b. ¿Cuál es la complejidad de la idea de tu compañero en el peor caso?
    c. ¿Qué le dirías a tu compañero al respecto de si su idea es una mejora a ***quickSort***? Argumenta.

3. En esta pregunta, un ABB es un árbol binario de búsqueda "básico", es decir, sin propiedades de balance; en cambio, un árbol rojo-negro es un árbol binario de búsqueda que además cumple las propiedades adicionales correspondientes:

    a. Encuentra una secuencia de claves —números enteros distintos— para insertar tanto en un ABB como en un árbol rojo-negro, tal que la altura del ABB resultante **sea menor** que la altura del árbol rojo-negro resultante; ambos árboles están inicialmente vacíos. Muestra, a través de una o más figuras, que tu secuencia efectivamente produce el efecto buscado.

    b. En un árbol rojo-negro inicialmente vacío, inserta la siguientes n claves en orden: **21, 3, 18, 15, 5, 10.** Para cada inserción, explica de manera breve y precisa, a través de una o más figuras, qué ocurre desde el punto de vista de las propiedades de un árbol rojo-negro y qué es necesario hacer en consecuencia antes de poder dar por finalizada la operación de inserción.

4. Con respecto a los árboles AVL.
    a. En clases vimos que para un árbol AVL de altura $\boldsymbol{h}$, el número mínimo de nodos (claves), $\boldsymbol{m(h)}$, cumple la recurrencia $\boldsymbol{m(h) = m(h-1) + m(h-2) + 1}$, y resolvimos esta recurrencia apoyándonos en las propiedades de la secuencia de Fibonacci.

       Pero también es posible resolver la recurrencia de otra forma: notamos que $\boldsymbol{m(h-1) > m(h-2)}$; por lo tanto, $\boldsymbol{m(h) > 2m(h-2)}$. Así, los primeros pasos del nuevo desarrollo son

       $$\boldsymbol{m(h) > 2m(h-2) > 4m(h-4) > 8m(h-6) > \ldots}$$

       Termina de resolver la recurrencia y encuentra la relación entre la altura h, y el número mínimo de nodos $\boldsymbol{m(h)}$, en notación $\boldsymbol{O}(\ldots)$, en un árbol AVL.

    b. Escribe tus dos apellidos, todo en letras mayúsculas y sin dejar espacios, y considera las primeras 6 letras distintas, de izquierda a derecha; por ejemplo, si tus apellidos son LEISER SON, entonces las letras a considerar son L E I S R O. Inserta esas 6 letras (las de tus apellidos) en ese mismo orden en un árbol AVL inicialmente vacío. Para cada inserción, explica de manera breve y precisa, a través de una o más figuras, qué ocurre desde el punto de vista de las propiedades de un árbol AVL y qué es necesario hacer en consecuencia antes de poder dar por finalizada la operación de inserción.