Algoritmos y Estructuras de Datos II - 17 de junio de 2015

Segundo Parcial

Alumno: .....

Siempre se debe explicar la solución, una respuesta correcta no es suficiente sino viene acompañada de una justificación que demuestre que la misma ha sido comprendida. Las explicaciones deben ser completas. La utilización de código c o de nivel de abstracción excesivamente bajo influirá negativamente. **Por favor, resolvé cada ejercicio en una hoja aparte,** para acelerar el proceso de corrección. **¡No olvides escribir claramente tu nombre en cada una de las hojas!** En los ejercicios con varios incisos, por favor, no resuelvas varios incisos simultáneamente, sino cada uno por separado (pero no hace falta que sea en hojas aparte).

1. Sea a un arreglo de $2^n - 1$ números naturales sin repeticiones ordenado de menor a mayor. Sea t un árbol binario de búsqueda inicialmente vacío. Utilizando la operación de inserción de un elemento en un árbol binario de búsqueda, ¿en qué orden podría insertar uno a uno los elementos del arreglo a en el árbol binario de búsqueda t de modo de que el árbol binario de búsqueda contenga al finalizar exactamente los elementos que inicialmente había en a, y esté balanceado? Luego de responder esa pregunta, escriba un algoritmo que construya efectivamente dicho árbol binario de búsqueda.

2. ¿Cuáles afirmaciones son verdaderas y cuáles falsas? Justificar.

   (a) Estas tres maneras de recorrer un ABB: pre-order, in-order y post-order; corresponden a recorridas DFS.

   (b) Al recorrer un grafo en DFS, si hay una arista de v_0 a v_1 seguro que se visita v_0 antes que v_1.

   (c) Al recorrer un grafo en BFS, si hay una arista de v_0 a v_1 seguro que se visita v_0 antes que v_1.

3. Se tienen las siguientes ciudades por las que se quiere pasar siguiendo el orden $0, 1, \ldots, n$ en el menor tiempo posible. Contamos con los tiempos $t_1, \ldots, t_n$, donde t_i es el tiempo medido en horas para llegar de la ciudad $i-1$ a la ciudad i. Se desea minimizar el tiempo total desde la hora de salida desde la ciudad de partida 0 hasta la hora de llegada a la ciudad de llegada n contando todas las horas intermedias, incluso si se ha estado detenido. Además solamente se permite viajar con luz diurna, esto es, entre las 8 horas y las 20 horas. Por lo que puede ser necesario pernoctar en ciudades intermedias. Se asume que ninguno de los trayectos entre ciudades consecutivas lleva más de 12 horas, es decir, que $t_i \leq 12$ para todo i. Dar un algoritmo que obtenga el menor tiempo T, medido en horas, necesario para visitar las n ciudades bajo estas condiciones y el conjunto C de ciudades en las cuales se tuvo que pernoctar.

4. Usted es un profesor de danza un tanto obsesivo y quiere que la próxima presentación del grupo salga óptima. El ballet está integrado por n hombres y n mujeres que deben bailar en parejas de un hombre y una mujer, todas las parejas simultáneamente en el escenario.

   El problema es que no todas las parejas hacen una buena pareja de baile. Por esa razón, a todas las parejas posibles de un hombre con una mujer (en total son n^2 parejas posibles) le ha asignado un puntaje, es decir, un número que es más alto cuando mejor es esa pareja de baile. Se trata de idear, escribir y explicar un algoritmo que utilice backtracking para decidir cuál es el mayor puntaje total. El puntaje total para una configuración de n parejas se obtiene sumando el puntaje de cada una de las n parejas de esa configuración. El algoritmo debe obtener el mayor puntaje total posible. Indicar luego cómo transformar el algoritmo (si es que hace falta) para que obtenga también la configuración óptima.

5. Para el problema de la moneda, se propone la siguiente solución. Se recuerda que se cuenta con suficientes (o infinitas) monedas de cada denominación $d_1, d_2, \ldots, d_n$ y se debe pagar de manera exacta el monto dado M. **No se asume que las denominaciones estén ordenadas.**

   Se cuenta con una solución que utiliza backtracking. La misma consiste en definir $m(i, j) =$ "menor número de monedas necesarias para pagar el monto j eligiendo entre las monedas de denominación $d_1, d_2, \ldots, d_i$" que da lugar a la siguiente definición recursiva:

   $$m(i,j) = \begin{cases} 0 & \text{si } j = 0 \\ 1 + \min\{m(k, j - d_k) | 1 \leq k \leq i \wedge d_k \leq j\} & \text{si } j > 0 \end{cases}$$

   Otra manera equivalente de escribir la última ecuación sería:

   $$m(i,j) = 1 + \min\{m(1, j - d_1),\ m(2, j - d_2),\ \ldots,\ m(i-2, j - d_{i-2}),\ m(i-1, j - d_{i-1}),\ m(i, j - d_i)\}$$

   en caso de que j sea mayor o igual a todas las denominaciones $d_1, d_2, \ldots, d_i$. En el caso general, deben omitirse los términos de la forma $m(k, j - d_k)$ si $j < d_k$. Por esa razón, en algunos casos puede que se esté calculando el mínimo de un conjunto vacío, que se asume que es infinito. También se asume que 1 más infinito es infinito.

   El ejercicio consiste en transformar **esta** solución en una que utilice programación dinámica. **No es válido recurrir a otra** solución al mismo problema. Sí es válido asumir que el tipo **nat** del pseudocódigo cuenta con un valor infinito con las propiedades deseadas.