Facultad de Informática

Ingeniería en Informática de Gestión

Estructuras de Datos y de la Información

Grupo A, Abril 2002

Ejercicios sobre árboles

Ejercicio 1 Especificar una operación que calcule la *frontera* de un árbol general, donde por frontera se entiende la lista formada por los elementos almacenados en las hojas del árbol tomados de izquierda a derecha.

Ejercicio 2 Extender la especificación de los árboles binarios con las operaciones siguientes:

- Calcular la talla de un árbol, definida como el número de nodos de la rama más larga.
- Calcular el número de nodos.
- Calcular el número de hojas.
- Reconocer si un árbol binario es completo.
- Reconocer si un árbol binario es casi completo.

Ejercicio 3 Demostrar que un árbol binario completo de talla $n \geq 1$ tiene 2^{n-1} hojas y un total de $2^n - 1$ nodos.

Ejercicio 4 Se dice que un árbol binario es *zurdo* si o bien (1) es el árbol vacío o una hoja, o bien (2) sus hijos izquierdo y derecho son ambos zurdos y más de la mitad de sus descendientes son descendientes izquierdos. Desarrollar un algoritmo recursivo eficiente para decidir si un árbol binario es o no zurdo.

Ejercicio 5 Dado un árbol binario, la *distancia* entre dos nodos es la longitud del único camino que los conecta, y el *diámetro* del árbol es la distancia máxima sobre todos los pares de nodos. Desarrollar un algoritmo de coste lineal con respecto al número de nodos del árbol para hallar el diámetro del un árbol binario dado.

Ejercicio 6 Desarrollar un algoritmo que reconstruya un árbol binario a partir de las dos listas que son sus recorridos en preorden e inorden, sabiendo que todos los elementos son distintos. Decir si es posible reconstruir el árbol unívocamente cuando las dos listas son sus recorridos en inorden y postorden. Lo mismo si las dos listas son sus recorridos en preorden y postorden.

Ejercicio 7 Suponemos disponible un tipo *personaje* con dos operaciones:

$esdragon : personaje \rightarrow bool$
$esprincesa : personaje \rightarrow bool$

que permiten averiguar si un personaje dado es un dragón o una princesa, respectivamente. Suponemos que ningún personaje es dragón y princesa a la vez, y que un personaje puede no ser ninguna de las dos cosas. Dado un árbol binario de personajes, se denominan nodos *accesibles* a aquellos nodos tales que a lo largo de la raíz hasta el nodo (ambos inclusive), no se encuentra ningún dragón.

- Especificar una operación que calcule el número de nodos accesibles de un árbol ocupados por una princesa.
- Programar una función iterativa de coste lineal que encuentre una princesa accesible lo más cerca posible de la raíz de un árbol dado a, suponiendo que algún nodo accesible de a contiene una princesa.

Ejercicio 8 Escribir un algoritmo que ordene una lista de elementos distintos utilizando como estructura auxiliar un árbol binario de búsqueda.

Ejercicio 9 Implementar los árboles binarios de búsqueda en cuyos nodos se guardan pares de la forma $\langle c, i \rangle$ donde c es una clave perteneciente al tipo *clave* que admite una relación de orden total e i es la información asociada a la clave perteneciente a un tipo *info*. La propiedad de ser árbol de búsqueda se establece en base al campo clave. Es necesario adaptar las operaciones vistas en clase.

Ejercicio 10 Suponiendo que el tipo de los elementos sobre el que se construyen los multiconjuntos admite una relación de orden total, implementar la especificación de los multiconjuntos en términos de árboles de búsqueda en cuyos nodos se guardan pares de elemento y multiplicidad de dicho elemento en el multiconjunto.

Ejercicio 11 El *problema de las concordancias* consiste en lo siguiente: dado un texto se trata de contar el número de veces que aparece en él cada palabra, y producir una lista ordenada alfabéticamente por palabras, donde cada palabra aparece acompañada del número de veces que ha aparecido en el texto. Suponemos que el texto a analizar viene dado como una lista de palabras, siendo *palabra* un tipo disponible con su orden. Se pide desarrollar un algoritmo que resuelva el problema con ayuda de un árbol de búsqueda.