Estructuras de Datos I

Ejercicios de árboles

Universidad Rey Juan Carlos

Ejercicio 1

- Fijar el árbol binario cuyos recorridos en preorden e inorden son las secuencias siguientes:
  - Preorden: L D A Y G K M S Q Z
  - Inorden: A D G Y K L M Q S Z

Ejercicio 2

- Represente gráficamente el árbol binario de búsqueda cuyos recorridos en preorden e inorden son las secuencias siguientes:
  - Preorden: 10, 3, 1, 5, 4, 8, 16, 20, 17, 21
  - Inorden: 1, 3, 4, 5, 8, 10, 16, 17, 20, 21
- A continuación, indicar cómo quedaría el árbol si se borra el nodo con clave 3. Describir el proceso de borrado de dicho nodo.

Ejercicio 3

a) Construir el árbol binario de búsqueda para la siguiente secuencia de elementos:

   [3,24,25,14,47,34,11,0,-2,40,20,7,5,-6]

b) Eliminar, del árbol anterior, las siguientes claves:

   [47,40,14,24,34]

Ejercicio 4

Se tiene un árbol binario de búsqueda estático simulando memoria dinámica en un array con la siguiente disposición inicial:

1		2		3		4		5		6		7		8		9		10	
Ana		Pedro		Cesar		Mario		Julia		Eva		Víctor		Raquel		Zoe		Jose	
0	0	1	3	3	6	5	7	10	0	3	8	0	9	0	1	0	0	0	0

cabArbol: 4

cabVacios: 2

a) Definir los tipos de datos y los TADs necesarios para resolver el problema que se plantea. Supóngase el tElemento de tipo string. ¿Cuántas hojas tiene el árbol en la situación inicial?, ¿por qué?

Ejercicio 4 (cont)

b) Devolver la secuencia de nombres que daría el recorrido preorden después de insertar el elemento “Ana”.

Ejercicio 5

Se pide un subprograma que sea capaz de sumar el valor de todos los nodos de un árbol binario. Se dispone de la siguiente función en el tElemento:

```
int Valor(tElemento e);
```

Ejercicio 6

Haciendo uso del TAD ÁrbolBinario y el TAD Lista vistos en clase, implemente el subprograma `PadresNoAbuelos` que dado un árbol binario devuelva una lista con los elementos que pertenecen a los nodos del árbol que sean padres, pero no abuelos. Asumimos que se dispone de la implementación de la función `esHoja`, que recibe un árbol y devuelve 0 si no es hoja y 1 si es nodo hoja. Se entiende que el tipo base del arbol binario y de la lista es el mismo `tElemento`.

Ejercicio 7

Considere las siguientes especificaciones algebraicas de operaciones sobre árboles ¿qué operaciones son Misterio y Misterio2? Explique su respuesta

OPERACIONES
Misterio: TipoArbolBin -> TipoLista
Misterio2: TipoArbolBin -> TipoLista
…
ECUACIONES
Misterio(CrearArbolBinVacio) = CrearVacia
Misterio(ConstruirArbolBin(i, r, d)) = SI EsArbolBinVacio(i) Y EsArbolBinVacio(d) →
Construir(r, CrearVacia)
| Concatenar(Misterio(i), Misterio(d)).

Misterio2(CrearArbolBinVacio) = CrearVacia
Misterio2(ConstruirArbolBin(i, r, d)) = InsertarFinal(r, Concatenar(Misterio2(i),
Misterio2(d)))

Ejercicio 8 (I)

Dada la pertinaz sequía que sufrimos actualmente, se desea tener control de la cantidad de agua que tienen los embalses en nuestro país, para tomar medidas respecto al riego y consumo humano. Será necesario consultar con frecuencia el porcentaje de capacidad de un determinado embalse para poder tomar las decisiones oportunas.

Para cada embalse se guardará la siguiente información: nombre del embalse, capacidad total y % de agua existente respecto a su capacidad total y se dispone una interfaz con las siguientes operaciones:

```
char *GetNombre(tEmbalse e);
float GetCapacidad(tEmbalse e);
float GetPorcentaje(tEmbalse e);
```

Se pide:
a) Definir en C una estructura que guarde la información antes mencionada sobre los embalses y que facilite la búsqueda de información sobre el estado de cualquiera de ellos.

Ejercicio 8 (y II)

b) Desarrollar en C una función que indique el porcentaje de agua de un determinado embalse. Esta función recibirá como parámetros la estructura que almacena la información de todos los embalses y el nombre del embalse (cadena de caracteres) del que se desea conocer su situación. El subprograma devolverá el porcentaje de agua existente.

c) ¿Qué complejidad tiene la operación del apartado b)? Justifique su respuesta.

Ejercicio 9 (I)

Se desea crear un sistema rápido para catalogar por edades novelas y conocer si son adecuadas para público infantil, mayores de 13 años o adultos, teniendo en cuenta una lista “negra” de palabras. Para ello se hace un análisis de las novelas, dependiendo de las palabras utilizadas y su frecuencia de aparición. El sistema recibe como entrada un fichero de texto ASCII con la novela, con posiblemente varios miles de palabras, y se organizan en memoria las palabras que contiene y sus frecuencias. Por ejemplo, con el texto "hola, adios, hola casa perro casita perro domingo", en memoria se guardarían las palabras y su frecuencia, y la salida ordenada alfabéticamente sería la siguiente:

adios 1
casa 1
casita 1
domingo 1
hola 2
perro 2

Se proporciona la operación ReadText que, dado el nombre del fichero de texto, devuelve un array sin orden con las palabras que contiene (un máximo de 20000 palabras, sin tener en cuenta signos de puntuación y otros caracteres que no formen parte de las palabras) y un valor con el número de palabras devueltas en el array:

```
Typedef char tWords[20000];

void ReadText(char *filename, twords *words, int *n);
```

Ejercicio 9 (y II)

A) Decidir qué estructura de datos de las estudiadas a lo largo del curso puede ser la más adecuada para resolver el problema para hacer comprobaciones rápidas de la existencia de palabras que un comité considera que no deben aparecer en novelas infantiles, aparecer poco en novelas para mayores de 13 años o que no importa que aparezcan en las novelas para adultos. Justificar la respuesta y definir los tipos necesarios.

B) Implementar las operaciones necesarias para hacer la carga de las palabras y su frecuencia, y luego mostrar el resultado por pantalla en orden alfabético. Asúmase accesible, ReadText.

C) Dada una lista negra de palabras desordenadas en una variable tWords del tipo anteriormente definido y una variable entera que describe el número de palabras que contiene la lista negra, mostrar en pantalla las palabras de la lista negra y el número de apariciones que tiene una novela cuya información se ha cargado en una estructura de datos creada en los apartados anteriores.