Universidad de Granada

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIA

ETSII Informática y de Telecomunicación, C/ Periodista Daniel Saucedo Aranda s/n- 18071- Granada (España)

Estructuras de Datos
Curso 2016-2017. Convocatoria de Julio
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas

1. (1 punto) Razonar **detalladamente** (en términos de eficiencia de las operaciones) qué TDA elegirías entre los que se indican para:
   (a) Implementar un **conjunto ordenado de enteros** (con funciones básicas de **insertar, borrar, y buscar**) de entre: un **ABB** (árbol binario de búsqueda no balanceado), un **APO** (árbol parcialmente ordenado implementado mediante un **heap**) y un v**ector ordenado.**
   (b) Implementar un **conjunto no ordenado de enteros** (con funciones básicas de **insertar, borrar y buscar**) de entre: un **AVL**, un **vector ordenado**, y un **APO** (árbol parcialmente ordenado implementado mediante un **heap**).

2. (2.5 puntos) Una lista **L** se dice que está ordenada de forma ascendente por índices (orden indexado) si la secuencia ***{L(indices(0)),L(indices(1)),...L(indices(indices.size()-1))}***. *(siendo L(i) el elemento en la posición i-ésima), es una secuencia ordenada de forma ascendente. Necesariamente* ***indices*** *debe ser una lista de enteros que indica una secuencia de índices en L. Implementa una función booleana que reciba L (L puede ser cualquier tipo de dato) y un índice I y devuelva true si L está ordenada de forma ascendente con respecto al índice.*
Ejemplo: Para L= {2.2, 7.1, 3.4, 9.0, 6.2, 5.1} e I={0, 2, 5, 4, 1, 3}, la función devolvería true.

3. (2.5 puntos) Implementa una función booleana que devuelva true si un árbol binario de enteros ab2 coincide con un subárbol de otro ab1 (deben coincidir también los valores de las etiquetas)

bool es_subarbol(const bintree<int> & ab1, const bintree<int> & ab2)

4.- (1.5 punto) Implementa un **TDA cola** usando **dos pilas**.

5.- (2.5 puntos) Se dispone de un TDA **indice** donde para cada palabra se almacenan los números de página en los que aparece en un libro. Se desea recorrer el índice considerando únicamente las ocurrencias en páginas impares de las palabras del contenedor. Para implementar el índice se ha seleccionado la siguiente estructura:

```
class indices{
    private:
       map<string, set<int> > datos;
    ……..
    ……..
}
```

Se pide implementar el TDA **indice::iterator** deseado, de forma que el **operator*** devuelva tanto la palabra como el número de página (que ha de ser necesariamente impar).
Además de los métodos de la clase iterador, se deben implementar las funciones begin() y end().

Tiempo: 3 horas

Universidad de Granada
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIA
ETSII Informática y de Telecomunicación, C/ Periodista Daniel Saucedo Aranda s/n- 18071- Granada (España)

Preguntas específicas para aquellos estudiantes sujetos a convocatoria adicional y evaluación única final, o que no hicieran las prácticas ni ejercicios de evaluación continua durante el cuatrimestre (4 puntos)

1.- (2 puntos) Sea el TDA editor que mantiene un conjunto de lineas de texto. Cada línea viene delimitada por un retorno de carro. Las operaciones que se pueden realizar son:

1. insertar una nueva linea de texto,
2. borrar la última línea de texto añadida (redo),
3. deshacer todo lo que se ha hecho desde un borrado previo (undo),
4. mostrar todas las líneas insertadas desde las más antigua hasta la más moderna.

Se pide:

1. dar una representación para el TDA editor,
2. implementar las funciones:
    1. Insertar una nueva línea de texto,
    2. borrar la última línea de texto,
    3. deshacer el último borrado,
    4. mostrar el contenido del editor desde la línea mas antigua a la linea mas reciente insertada.

NOTA: Se puede deshacer consecutivamente varios borrados que se hayan podido realizar previamente.

2.- (2 puntos) Dado un árbol binario, implementar una función que devuelva en una lista los elementos de un determinado nivel en el árbol, sin usar iteradores:

list<T> nivel(const bintree<T> &arb, int l)

Duración 1 hora