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 2013-2014. Convocatoria de Septiembre
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas

1. (1.5 puntos) Se desea diseñar un TDA, que llamaremos **colec-ord** que representa a una colección ordenada (de menor a mayor) de enteros y donde las únicas operaciones permitidas son **insertar**, **borrar-máximo**, **recuperar** y **cuantos-dentro-de-rango**. La operaciones borrar-máximo y recuperar, respectivamente borran el elemento máximo y recuperan un entero en la colección, y la operación cuantos-dentro-de-rango tiene como parámetros un rango de valores enteros [a,b] (a<b) y devuelve un valor entero indicando **cuantos** de los enteros de la colección son mayores o iguales que **a** y menores o iguales que **b**. Analizar la eficiencia de las operaciones si se usa (a) vector de la STL (b) lista de la STL (c) un ABB, (d) un AVL. ¿Qué implementación escogerías para el TDA?

2. (2.5 puntos) Se desea construir un traductor de un idioma origen a un idioma destino. Una palabra en el idioma origen puede tener más de un traducción en el idioma destino.
   - Dar una representación para el TDA Traductor
   - Implementar la función **insertar** que añade una palabra del idioma origen junto con las traducciones en el idioma destino.
   - Implementar la función c**onsultar** que obtiene las traducciones de una palabra en el idioma destino.
   - Implementar la clase **iterador** dentro de la clase Traductor para poder iterar sobre todas las palabras.

3. (2 puntos) Construir una función que permita "postintercalar" dos listas (intercalar alternativamente todos los nodos que las integran de final a principio), con los siguientes requisitos:
   (a) Hay que comprobar que las dos listas no son vacías
   (b) Se empieza por el primer nodo de la primera lista
   (c) La segunda lista contendrá el resultado final y la primera quedará vacía.
   (d) Si una de ellas tiene un menor número de nodos que la otra, el exceso de nodos se incorporará a la lista resultante.

Ejemplo 1:	Ejemplo 2:
Antes de invocar al metodo	*Antes de invocar al metodo*
Primera lista : (100,200)	Primera lista : (x,y,z)
Segunda lista: (1,2,3,4,5,6)	Segunda lista: (a,b,c,d,e,f)
Despues de invocar al metodo	*Despues de invocar al metodo*
Lista segunda: (1,2,3,4,200,5,100,6)	Lista segunda: (a,b,c,z,d,y,e,x,f)
Lista primera: vacia	Lista primera: vacia

4. (2 puntos) Implementa la función

   bintree<T>::node siguiente_nodo_nivel(const bintree<T>::node &n, const bintree<T> &arb)

   que dado un nodo n de un árbol arb, devuelve el siguiente nodo que está en ese mismo nivel del árbol.

5. (2 puntos) Un "**q-APO**" es una estructura jerárquica que permite realizar las operaciones *eliminar-minimo* e *insertar* en un tiempo O(log_2(n)), y que tiene como propiedad fundamental que para cualquier nodo X la clave almacenada en X es **menor** que la del hijo izquierda de X y esta a su vez **menor** que la del hijo derecha de X, siendo el árbol binario y estando las hojas empujadas a la izquierda.
   Diseñar una función para insertar un nuevo nodo en la estructura. Aplicarlo a la construcción de un q-APO con las claves {29, 24, 11, 15, 9, 14, 4, 17, 22, 31, 3, 16}.

Tiempo: 3 horas