Estructuras de Datos

Práctica 4. Árboles avl

Sesiones de prácticas: 2

Objetivos

Construir un árbol avl y manejarlo sin recursividad

Ejercicio propuesto

En esta práctica vamos a trabajar con grandes cantidades de datos que necesitan realizar búsquedas eficientes, el caso típico en el que se deben manejar árboles AVL. En este caso usaremos un fichero de más de 40.000 códigos postales de EEUU (fichero adjunto databasde_codPost.tar.gz). El fichero que contiene está en formato csv, en él aparecen campos como el código postal, el nombre de la ciudad, el estado al que pertenece e incluso las coordenadas terrestres en el mapa.

Lo que se pretende es cargar todos estos datos en memoria de modo que el árbol utilice como clave el **nombre de la ciudad**. Como una ciudad puede tener más de un código postal, todos estos registros quedarán guardados en una lista enlazada asociada a cada nodo del árbol AVL. La lista enlazada puede ser la misma utilizada en la Práctica 3.

Como el árbol puede ser bastante grande, ante los posibles desbordamientos de la pila, se va a optar por trabajar con funciones no recursivas tanto para la búsqueda como para la inserción de datos.

Los pasos a seguir en la práctica son los siguientes:

- Crear la clase Avl<T> con las operaciones típicas exceptuando el borrado:

```
template <typename T> class Avl {
    Nodo<T> *raiz;

    void rotdecha(Nodo<T>* &p);
    void rotizqda(Nodo<T>* &p);
public:
   Avl();
   ~Avl();
   bool insertarNR(const T &dato);
    T *busquedaNR(const T &dato);
};
```

- Donde:

```
T *Avl<T>::busquedaNR(const T &dato);
```

  localiza el dato que corresponde con la clave introducida de forma iterativa (no recursiva).

- Y la función:

```
bool Avl<T>::insertarNR(const T &dato);
```

  inserta el dato también de forma no recursiva utilizando una pila. Se puede implementar la clase Pila<T> a partir de la clase Lista<T> mediante herencia , aunque en este caso también puede utilizarse dicha lista directamente utilizando las operaciones de insertar y borrar por el comienzo de la lista. La operación *insertarNR()* devuelve *true* si se ha conseguido realizar la inserción (no existía previamente el dato en el árbol) y *false* en caso contrario.

- Leer el fichero adjunto y cargar cada uno de los registros en un objeto de la clase *CodigoPost* (11 atributos). Estos objetos deberán insertarse convenientemente en la estructura AVL de listas descrita anteriormente. Estos objetos serán parte de las listas asociadas a los nodos del árbol antes descrito. La definición de esta estructura de datos será:

```
Avl<StructPost> codigosPostales;
```

siendo StructPost:

```
struct StructPost{
    string ciudad;
    Lista<CodigoPost> codigos;
    bool operator<(const StructPost &cp){
       return (ciudad < cp.ciudad);
    }
};
```

Aplicación

La prueba del ejercicio consistirá en obtener todos los códigos postales (y el resto de la información) para una ciudad dada como entrada.