Documentación Práctica 2 ED

Alumno: Adrián Bennasar Polzin**, DNI:** 41 54 23 28 G**.**

Grupo: 1.

Las estructuras utilizadas en esta práctica son:

1. Mapping implementado con un array (tabla de frecuencias).
2. Árbol binario.
3. Heap.

En este documento se presenta una breve descripción junto con un diagrama para cada una de las estructuras.

Mapping implementado con un array (tabla de frecuencias)

El diagrama de esta estructura se puede visualizar en el siguiente enlace:

mappingArray

Esta estructura consiste en un conjunto construido a través de un record formado por 2 arrays, y acompañado de un iterador.

Estos arrays están ambos indexados con los caracteres que usamos habitualmente, es decir, caracteres de la ‘a’ a la ‘z’ asi como caracteres especiales como ‘ ‘, ‘?’, etc. Es decir, son los llamados arrays indexados por claves, estudiados en esta asignatura.

El array que recibe el nombre de **existencia** se utiliza para indicar a través de un booleano si un carácter dado existe o no en el conjunto.

El array con el nombre **contenido** se utiliza para indicar la frecuencia de aparación de un carácter dado, a través de un tipo integer.

Finalmente tenemos el iterador, para iterar sobre la estructura realizando funcionalidades como puede ser un recorrido o una busqueda.

Árbol binario

El diagrama de esta estructura se puede visualizar en el siguiente enlace:

árbolBinario

En esta práctica se utilizan árboles binarios de un solo nodo.

Esta estructura es muy simple: consiste en un puntero que apunta a la raiz del arbol, la cual contiene el unico nodo de este.

Este único nodo del árbol, contiene un tipo de datos definido por nosotros, llamado también nodo. Este tipo de dato es un record formado por dos tipos base, un tipo carácter para representar una clave, y un tipo entero para representar la frecuencia.

Podemos crear entonces árboles de un solo nodo que contienen parejas clave-valor. Por cada carácter de la tabla de frecuencias, creamos un árbol con la pareja clave-valor y lo insertamos en otra estructura de datos, **Heap**, la cual se documentará a continuación.

Heap(cola de prioridad)

El diagrama de esta estructura se puede visualizar en el siguiente enlace:

Heap(cola de prioridad)

Esta es la estructura donde insertamos y extraemos árboles de un solo nodo.

Consiste en un array donde tanto los elementos de este como el tamaño tienen la propiedad **generic**, es decir, dependiendo de la implementación de la estructura el programador podra elegir el tamaño y qué elementos quiere guardar en el array.

Por lo tanto aplicado a esta práctica, a la hora de instanciar el paquete indicamos que el array contendrá arboles, y una tamaño que consideremos conveniente para el objetivo de la práctica.

Al ser una cola de prioridad, también cuenta con dos operaciones **generic,** menor y major, con las que indicamos qué significa que un arbol sea más prioritario que otro. Una vez hecho esto, al añadir o eliminar arboles en el array, este se ordenará automáticamente con el criterio especificado.