UNIVERSIDAD DE ALMERÍA

Estructuras de datos y Algoritmo I

Práctica 02: ejercicio 03

Curso: 2º Grado en Ingeniería Informática
Grupo docente: B1 **Grupo**: GTA3
Nombre y Apellidos:
Jefferson Max Tomalá Villarreal

Ejercicio 3. Árbol binario de búsqueda AVL. Corrector ortográfico.

El criterio de *similitud* entre parejas de palabras que vamos a seguir en este ejercicio se basará en la conocida *distancia de Levenshtein* http://es.wikipedia.org/wiki/Distancia_de_Levenshtein).

En este punto surgen algunas cuestiones que tendréis que pensar antes de implementar la solución:

Por ejemplo, la distancia de Levenshtein entre "casa" y "calle" es de 3 porque se necesitan al menos tres ediciones elementales para cambiar uno en el otro.

1. casa → cala (sustitución de 's' por 'l')
2. cala → calla (inserción de 'l' entre 'l' y 'a')
3. calla → calle (sustitución de 'a' por 'e')

1. **¿Cuál es la idea básica de este método de similitud?**

La distancia Levenshtein se basa en la similitud que existe entre dos palabras.

2. **¿Cómo utilizaremos esta distancia de edición (edit distance, como también se le conoce a la distancia de Levenshtein) para recuperar del AVL las n palabras más similares?**

Para encontrar estas palabras más similares devolverá una lista de palabras con menor distancia Levenshtein es decir palabras que tenemos que hacer pocos cambios para convertir en la palabra consultada.

3. **¿Qué pasaría si el diccionario lo implementásemos mediante un BSTree, en lugar de con un AVLTree?**

La diferencia entre BSTree y AVLTree es que el segundo tiene una propiedad de balanceado es decir tiene un factor de equilibrio para el cual la diferencia entre las alturas de los subárboles de cada uno de sus nodos es, como mucho 1.

El árbol estará balanceado de modo que al recurrir a las operaciones, el orden de complejidad será de promedio el mismo Θ(log n), caso que en BSTree tenderá a Θ(n).