Auxiliar 9 - Más diccionarios

Profesores: Jérémy Barbay
Patricio Poblete, Nelson Baloian
Auxiliares: Felipe Lizama, Franco Sanguineti,
Matías Ramírez, Sven Reisenegger.

P1. Skip List

(a) Muestre el resultado de insertar las llaves$1, 4, 10, 3, 5$ en una `skiplist` vacía, considerando los lanzamientos de monedas $CCSCCCSSSS$, en donde C (cara) implica hacer crecer el nivel del nodo, y S (sello) implica terminarlo.

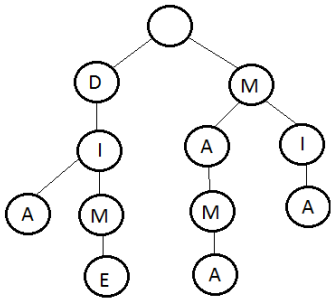
(b) ¿Habría sido distinto el resultado si las llaves hubieran sido insertadas en un orden distinto, pero usando la misma secuencia de lanzamientos de moneda aleatorios?

(c) ¿Cuántos pasos horizontales se realizan cuando se busca la llave 5? Muestre el camino de búsqueda para esta llave en su `skiplist`.

(d) Describa todo los necesarios para eliminar la llave 4 de la `skipList`.

P2. Tries

Un *trie* (o árbol digital) es un tipo de árbol de búsqueda (no necesariamente binaria), que tiene la particularidad de no guardar valores *en* sus nodos, sino que la información está dada por la posición que tiene un nodo dentro del árbol. La búsqueda se realiza desde la raíz, eligiendo la i-ésima arista a recorrer según i-ésimo caracter del valor que se desea buscar. Note que la búsqueda tarda O(largo de la palabra buscada).



(a) Inserte en un *trie* vacío las palabras `PANADERÍA`, `PAN`, `PANADERO`, `PANCITO`, `PALA`, `PALABRERÍA`, `PALABRA`. ¿Cuál es el prefijo más largo que se repite?

(b) Programe una función `buscar_prefijo` que reciba un nodo a un *trie* e imprima todas las palabras que comiencen con dicho prefijo.

.

P3. Ordenando con diccionarios

Considere un nuevo TDA `Diccionario Ordenado` que soporta los siguientes métodos:

- `insert(x)`: Inserta un elemento x en el diccionario ordenado.
- `sucesor(x)`: Retorna el mínimo de los elementos mayores a x, donde x puede ser $-\infty$, una clave externa mas pequeña que todas las claves del diccionario. Si es que no hay ningún elemento mayor, retorna `None`.

Suponga que tiene acceso a una clase `DiccionarioOrdenado` que implementa este TDA y que tiene un constructor vacío. Escriba una función `ordenar_via_diccionario` que reciba un arreglo desordenado de n enteros, y utilice la clase `DiccionarioOrdenado` para retornar un nuevo arreglo con los elementos ordenados sin modificar el arreglo original.

Incluya un análisis de tiempo en función del tiempo $\tau_i(n)$ de la funcion `insert(x)` y del tiempo $\tau_s(n)$ de la función `sucesor(x)` en la estructura de datos implementando el TDA `DiccionarioOrdenado`.