Primera Entrega - Proyecto de Clase

Laura Natalia Ballesteros Gualdrón - 2221650
Nicol Alexa Rodriguez Alfonso - 2240071
Héctor Alirio Valdeleón Millán - 2230063
Juan David Rey Ardila - 2210080

Debido al funcionamiento de las listas enlazadas, y lo poco eficiente que estas resultan para realizar búsquedas - debido a no tener acceso aleatorio a los elementos dentro de ella-, decidimos implementar dos scripts que solucionan el problema de ordenamiento y búsqueda dentro de una lista doblemente enlazada.

script1.py

El primer script contiene todas las implementaciones propuestas en el moodle.
Para el método de búsqueda utilizamos merge sort, y a partir de ahí, búsqueda secuencial, tanto para el método de conteo, como para el de impresión de la lista, que es lo más óptimo en este caso.
Cabe resaltar que el algoritmo de merge sort para listas enlazadas toma la forma O(logn), y el de búsqueda lineal O(n).

skip-list.py

El segundo script es un paso adelante en tiempos de ejecución y funciona bajo la estructura skip-list. Esta estructura se asemeja a la del metro de varias ciudades del mundo, y funciona por niveles, con la cabeza en la esquina superior izquierda y con valor menos infinito por defecto.

Para la implementación utilizamos 4 apuntadores por nodo, con todos los nodos en el primer nivel (nivel 0) y en los niveles superiores algunos de ellos, escogidos bajo aleatoriedad dentro del método insertar.

¿ Cómo funciona el algoritmo?
Insertar:

1. **Buscar(x)** y encontrar dónde debería estar posicionado en la lista con nivel 0.
2. Insertar x en la lista con nivel 0 para cumplir con la invariante que nos advierte de la existencia de todos los nodos en el primer nivel.
3. Lanzar una moneda. Mientras el resultado sea cara, promover el nodo a un nivel superior manteniendo la cabeza como extremo izquierdo en el nuevo nivel. De lo contrario, detener el procedimiento. (Este paso lo implementamos mediante la librería random de python)

Eliminar:

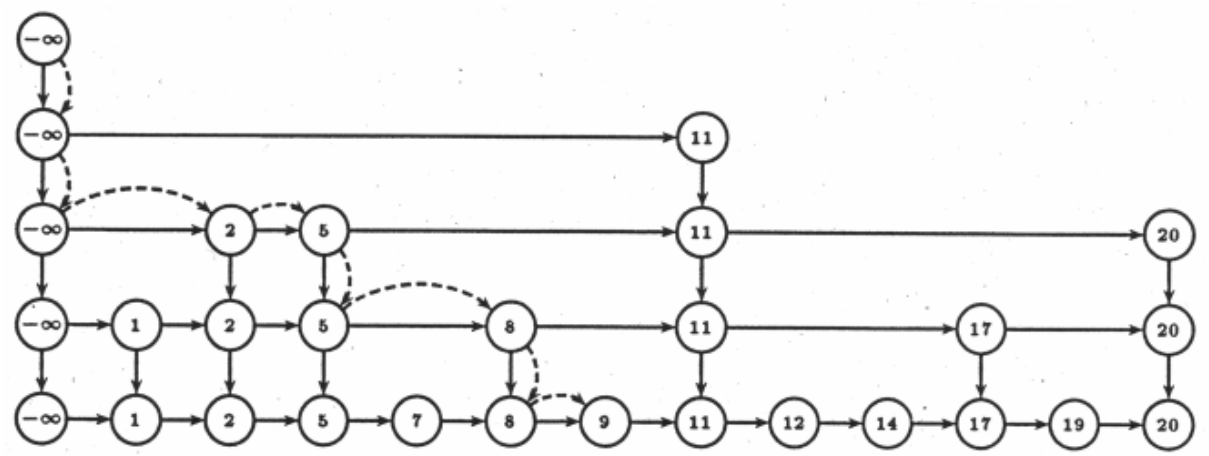
1. **Buscar(x)** y encontrar su posición.
2. Si x está en primer nivel. Desvincular sus apuntadores.
3. Si x está en un nivel superior. Desvincular sus apuntadores desde arriba hasta el nivel 0 o primer nivel.

Buscar:

1. Caminar la lista desde la esquina superior izquierda. Último nivel. Hasta encontrar un nodo con valor superior al que queremos buscar.
2. Bajar de nivel.
3. Seguir caminando hasta encontrar x o un nodo mayor que x.

Por defecto, el método de búsqueda retorna el nodo encontrado sea cual sea su nivel, sin embargo, esto no sucede cuando el nodo no se encuentra en la lista, y es ahí en donde el método insertar basa su funcionamiento, ya que si el nodo no se encuentra, el método de búsqueda bajará hasta el primer nivel o nivel 0 y devolverá la posición en donde debería estar, para luego realizar su inserción. Esto por defecto sucede en el nivel 0, puesto que si un nodo no está en un nivel superior, no estará tampoco en el nivel 0, y el método de búsqueda llegará hasta este nivel para encontrar la posición correcta.

Skip List se construye bajo aleatoriedad y aunque no es exactamente igual a un árbol binario, toma una forma parecida a un árbol. Sus métodos insertar, eliminar, y buscar, toman la forma O(logn).



Recursos:
https://ocw.mit.edu/courses/6-046j-introduction-to-algorithms-sma-5503-fall-2005/resources/lecture-12-skip-lists/

https://www.osa.fu-berlin.de/bioinformatics_msc/en/exemplary_tasks/informatics_algorithms/index.html