LinkedList - Método Search

→ Este método nos sirve para buscar un elemento dentro de la LinkedList según la posición que recibe como parámetro.

→ Lo primero que hará será verificar si la LinkedList está vacía o no. Si está vacía (no contiene ningún nodo/elemento) entonces retornará un valor False.

Si NO está vacía, entonces tomará el primer elemento (self.first) y lo guardará en una variable llamada 'current'. También creará una variable contador llamada 'count' cuyo valor será 0.

Luego, verificará con un ciclo while la existencia de current y mientras éste exista hará lo siguiente:

Verificará si la posición (index) introducida es igual al valor del contador (count), y si es así, retornará un valor verdadero. Si NO es así, entonces, current tomará el valor del nodo que le sigue (current.next), y a la variable count se le sumará 1. Esto se repetirá hasta que current

Sea igual a None. Si al finalizar el ciclo while count nunca fue igual a la posición (index), entonces significa que no hay ningún elemento en dicha posición. Por lo tanto, el método search retornará un valor False y terminará.

LinkedList - Método Replace.

→ Este método nos sirve para modificar un elemento dentro de la LinkedList según la posición que recibe como parámetro.

→ Lo primero de todo es que verificará si la posición (index) ingresada es realmente un número (int). Si no lo es, entonces retornará un valor False y termirá su ejecución.

De lo contrario, verificará si la LinkedList está vacía. Si está vacía entonces retornará un valor False. Si tiene al menos un elemento entonces guardará al primer elemento de la LinkedList en una variable llamada 'current' y creará otra variable llamada count cuyo valor será igual a 0

Luego, verificará con un ciclo while la existencia de current y mientras este exista hará lo siguiente.

Verificará si la posición (index) es igual al valor del contador (count), y si es así, el valor de current será modificado por el objeto nuevo que introducimos como parámetro de la función, y retornará un valor True.
Si NO es así, entonces, current tomará el valor del nodo que le sigue (current.next) y a la variable count se le sumará 1 en cada iteración.

Esto se repetirá hasta que current sea igual a None. Si al finalizar el ciclo while, count nunca fue igual a la posición (index), entonces significa que no hay ningún elemento en dicha posición. Por lo tanto, retornará un valor False y terminará su ejecución.

Compare - Método Compare

→ Este método nos sirve para comparar nodos, específicamente su valor y retornar valores enteros que sí sean capaces de decirnos qué nodo irá antes o después del otro.

→ Primeramente, verificara si los objetos introducidos como parámetros son de tipo nodo, y si es así, guardará su valor (obj.value) en el objeto obj

Luego, transformará el contenido de las variables obj a cadena (string) por si alguno de los valores eran enteros.

Luego, obtendrá la longitud de la cadena más pequeña de las dos, utilizando un método llamado 'min'

Después, comenzará un ciclo for desde cero hasta un número menor que la longitud más pequeña (min), en el ejecutará lo siguiente:

Verificará si el caracter número 'i' del objeto1 (obj1) es menor que el caracter número 'i' del objeto 2 (obj2) según el índice que tenga dicho caracter en el abecedario (alphabet). Si es así, entonces retornará -1.
También verificará lo contrario, si el caracter i de obj1 es mayor al de obj2. Si es así, retornará 1.

Si al final del ciclo for, los caracteres fueron iguales, entonces verificará la longitud de las cadenas:

Si la longitud de obj1 es menor que la longitud de obj2, entonces retornará -1.

Si en cambio, la longitud de obj1 es mayor a la longitud de obj2, entonces retornará 1.

Si al final nada de eso se cumple, retornará 0 y terminará su ejecución.

✓ Agregar:

- Si no hay elementos: Se agrega uno nuevo
- Si hay: Saber en la posición que desea agregar
  - Si es en la posicion 'cero'
    - Guardamos en una variable el primero y despues agregamos el nuevo en el primero
    - El siguiente del primero (el elemento agregado) será igual al guardado en la variable
  - Si es cualquier otra posición
    - Tenemos el anterior y el actual
    - Recorremos la lista; comparando la posicion en que queremos agregar incrementando el anterior y el actual, si lo hallamos la posición:
      - El siguiente del anterior será igual al nuevo nelemento
      - El siguiente del siguiente del anterior será igual al actual
    - Si no encontramos la posicion se agregará despues del ultimo elemento.

✓ Eliminar:

- Si no hay elementos: no hace nada
- Si hay: Debemos saber la posición a eliminar
  - Si la posición es la primera en la lista
    - Guardamos en una variable el siguiente del primero y ese será el primer elemento de la lista
  - Si es cualquier otra posición
    - Guardamos el anterior en una variable y el siguiente del anterior en otra
    - Tendremos un contador para comparar la posición
    - Recorremos la lista
      - Nos preguntamos si el contador es igual a la posición dada
        - Si es igual: el siguiente es igual al siguiente de ese elemento y el siguiente del anterior es igual al siguiente
        - Si no es igual:
          - el anterior es igual al siguiente
          - el siguiente es igual al siguiente del siguiente

Tabla Asscii

- Obtener los datos en una lista (vector, matriz)

  -> Recorremos los elementos en el catalogo (lista del catalogo)

  -> Para cada elemento Movie obtenemos sus Valores (nombre, Duración, Categoria, etc...)
  -> los Valores extraidos se agregan al vector
  -> El vector se agrego a la matriz.
  :
  y asi hasta terminar el recorrido

  -> Retorna la matriz con todos los elementos que contendra la tabla.

- Obtener los espacios correspondientes a cada columna.

  -> Asignamos Variables inizializado en 0 para almacenar el maximo de tamaño de los cadenas de texto de cada elemento de la matriz en la posición i

  -> Comparamos si el valor de la matriz es mayor que al valor de las variables locales entonces el nuevo valor maximo seria este y asi hasta recorrer todos los elementos
  -> retorna el maximo len de las cadenas

Luego para cada elemento en el vector, matriz restamos el valor maximo - Valor del actual len y eso nos dara como resultado los espacios en cada columna.

- Crear la tabla
  - Obtenemos la data en la matriz
  - Hacemos el recorrido para obtener el Valor del vector en la posicion i
    - -> Asignamos espacios encontrados previamente
    - -> Se guarda el contenido en un string
  - Retorna el string con los elementos de la tabla.

- Mostrar la tabla:
  - -> Obtenemos el string que contiene todos los elementos en la tabla
  - -> Hacer set del contenido al QLexedit

-> Mostrar Árbol:
- Recorremos cada elemento del árbol
- En cada elemento agregamos su nodo a la instancia de graph de NetworkX
- Agregamos sus Aristas
- Generamos la imagen

Convertir de LinkedLis a BST

Este metodo convertira una lista enlazada a un árbol binario de busqueda.

- Para cada elemento en la lista enlazado:

  Si este elemento es uno Movie entonces
  - Se extra su valor. y se guarda en una variable current.

  - Mientras exista current.
    Cada elemento se va agregando a una instancia del BST.

  - Retorna el BST con todos los elementos agregados

Convertir Tiempo a segundo.

- Validamos que el formato de tiempo ingresado por el usuario sea Valido.

  → hacemos us split ":"
  Si el tamaño del list resultante al hacer split es == 3
  - Si al hacer un casteo a entero de cada elemento no hay ningun error (Try excepl) entonces retorna True en caso contrario retorna False

- Si al Validar retorna True

  - hacemos split ":" del formato
    Convertimos el elemento [0] a segundos
    Convertimos el elemento [1] a segundos
    Segundos ya esta almacenado en [2]

    Sumamos todos los segundos

    retorna la suma del tiempo total en segundos

Guardar en memoria.

- Permitira recobrar los elementos agregados a la lista

→ Guardar

→ Obtenemos el contenido de cada elemento en la lista enlazada.

Para cada elemento en la lista enlazada
→ agregamos a un diccionario cada valor con su llave correspondiente

Luego ese Valor a diccionario se agregara a un diccionario "General" con su respectiva clave (un valor numerico)

- Luego escribimos ese diccionario en un archivo .json en la ruta que queramos dentro del equipo.

Obtener de memoria.

- Abrimos el archivo JSON obtenida o generado en nuestro equipo
- Para cada llave desde 0, hasta el ultimo elemento
  - Obtenemos la data del dic con llave i cada elemento de esa llave i se agrega o se genera un obj Movie
    - Ese objeto movie se agrega a la lista enlazado
    - Se retorna la lista enlazada