Algoritmos y Estructuras de Datos

Unidad Temática 3

Práctico Individual 7

19 de abril de 2025

Santiago Blanco Canaparro
Profesor: Sebastián Torres
Grupo I2M2

---

1. Ejercicio #1 ()

→ **Lenguaje natural**:
Se tienen los conjuntos ordenados A y B (Listas Enlazadas) y se debe realizar la operación UNION entre ambos resultando en un conjunto C.

```
Defino conjunto C vacío
Creo dos referencias y las apunto al inicio de ambos conjuntos

Mientras ambas referencias no sean nulas
  Si el elemento de A es menor que el de B
    Agregar el elemento de A al conjunto C
    Avanzo al proximo elemento de A
  Fin Si
  Si el elemento de B es menor que el de A
    Agregar el elemento de B al conjunto C
    Avanzo al proximo elemento de B
  Fin Si
  Sino (si son iguales)
    Agrego el elemento una sola vez (que no haya duplicados)
    Avanzo al proximo elemento de ambos
  Fin Sino

  Si un conjunto se terminó antes que el otro
    Agrego los elementos restantes
  Fin Si

Retornar cojunto C
```

Precondiciones

- Los conjuntos A y B que se reciben están ordenados
- Los conjuntos A y B que se reciben tienen elementos comparables

Postcondiciones

- Se retorna una lista enlazada C
- La lista C contiene todos los elementos de A y B

- La lista C no contiene elementos repetidos

Se tienen los conjuntos A y B (Listas Enlazadas) y se debe realizar la operación INTERSECCION entre ambos resultando en un conjunto C.

```
Defino conjunto C vacío
   Creo dos referencias y las apunto al inicio de ambos conjuntos

   Mientras ambas referencias no sean nulas
     Si son iguales
       Agrego el elemento que está en ambos
       Avanzo al proximo elemento de ambos
     Fin Si
     Si el elemento de A es menor que el de B
       Avanzo al proximo elemento de A
     Fin Si
     Si el elemento de B es menor que el de A
       Avanzo al proximo elemento de B
     Fin Si

   Retornar conjunto C
```

Precondiciones

- Los conjuntos A y B que se reciben están ordenados
- Los conjuntos A y B que se reciben tienen elementos comparables

Postcondiciones

- Se retorna una lista enlazada C
- La lista C contiene todos los elementos x tal que $x \in C \iff x \in A \land x \in B$
- La lista C no contiene elementos repetidos

→ **Pseudocódigo**

```
union(conjuntoB: Lista Enlazada) -> Lista Enlazada
  COMIENZO
    Lista conjuntoC <- Nueva lista // O(1)

    actualA <- primero // O(1)
    actualB <- conjuntoB.primero // O(1)

    // Mientras ambos conjuntos tengan elementos
    // O(n)
    MIENTRAS (actualA != nulo Y actualB != nulo) HACER
        comparacion <- comparar(actualA, actualB) // O(1)
```

```
        SI comparacion < 0 ENTONCES
            resultado.insertar(actualA.valor, actualA.etiqueta)// O(1)
            actualA <- actualA.siguiente // O(1)

        SINO SI comparacion > 0 ENTONCES
            resultado.insertar(actualB.valor, actualB.etiqueta)// O(1)
            actualB <- actualB.siguiente // O(1)

        SINO
            resultado.insertar(actualA.valor, actualA.etiqueta)// O(1)
            actualA <- actualA.siguiente // O(1)
            actualB <- actualB.siguiente // O(1)
        FIN SI
    FIN MIENTRAS

    // AGREGA LOS ELEMENTOS RESTANTES

    // O(n)
    MIENTRAS actualA != nulo HACER
        resultado.insertar(actualA.valor, actualA.etiqueta)
        actualA <- actualA.siguiente
    FIN MIENTRAS

    // O(n)
    MIENTRAS actualB != nulo HACER
        resultado.insertar(actualB.valor, actualA.etiqueta)
        actualB <- actualB.siguiente
    FIN MIENTRAS

    RETORNAR resultado
  FIN
```

Análisis del tiempo de ejecución

Nota: dentro del bucle Mientras se referencian dos métodos propios del TDA Lista

- insertar() ->Es de orden O(1) porque existe una referencia al último nodo en la lista.
- comparar() ->Es de orden O(1) porque compara entre objetos comparables que tengan implementados el método compareTo().

Total: O(1) + O(1) + O(1) + [O(n) * [O(1)*8]] + O(n) + O(n) = $O(n)$

```
 interseccion(conjuntoB: Lista Enlazada) -> Lista Enlazada
  COMIENZO
    Lista conjuntoC <- Nueva lista  // O(1)

    actualA <- primero // O(1)
    actualB <- conjuntoB.primero // O(1)

    // O(n)
    MIENTRAS (actualA != nulo Y actualB != nulo) HACER
        comparacion <- comparar(actualA, actualB)  // O(1)

        SI comparacion == 0 ENTONCES
            conjuntoC.insertar(actualA.valor, actualA.etiqueta) // O(1)
            actualA <- actualA.siguiente  // O(1)
            actualB <- actualB.siguiente // O(1)

        SINO SI comparacion < 0 ENTONCES
            actualA <- actualA.siguiente  // O(1)

        SINO
            actualB <- actualB.siguiente // O(1)
        FIN SI
    FIN MIENTRAS

    RETORNAR conjuntoC // O(1)
FIN
```

Análisis del tiempo de ejecución

Nota: dentro del bucle Mientras se referencian dos métodos propios del TDA Lista

- comparar() ->Es de orden O(1) porque compara entre objetos comparables que tengan implementados el método compareTo().
- insertar() ->Es de orden O(1) porque existe una referencia al último nodo en la lista.

Total: O(1) + O(1) + O(1) + [O(n) * [O(1)*6]] = $O(n)$

Nota: La idea de los algoritmos la tomé del libro 'Estructuras de Datos y Algoritmos' de Alfred Aho.