
DEVZ
COMMUNITY

SESIÓN 2

Aprendiendo Linked Lists.

D{VZ

DEVZ

Linked Lists

Estructura de datos que representa una **secuencia o colección *lineal* de nodos**. En español se les conoce como **Lista Enlazada**

A diferencia de un Array, una Linked List **no** provee acceso a nodos en tiempo constante O(1), no hay índices.

DEVZ

Linked Lists

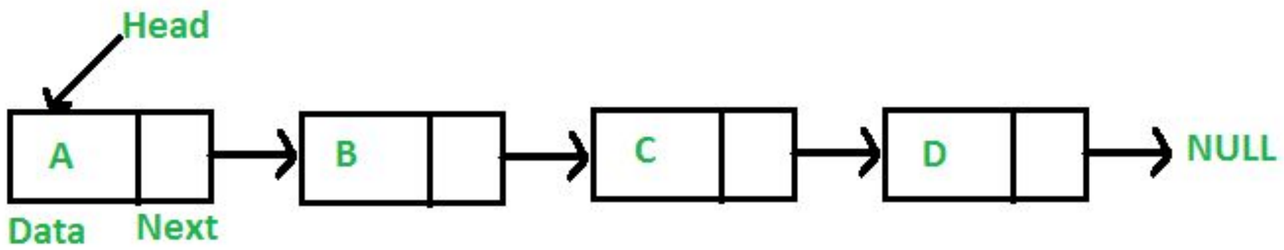
Una referencia a un LinkedList se obtiene con un **nodo** inicial, generalmente llamado cabeza o **head**. Wikipedia le llama **Nodo Centinela** o **Puntero Cabeza**

Al trabajar con una Linked List, solo se tiene acceso a **un nodo a la vez.**

La lista termina cuando el último elemento tiene una referencia **nula (null)** al siguiente nodo.

Linked Lists

En un **Singly Linked Lists**, cada nodo tiene una referencia al siguiente nodo.



https://www.geeksforgeeks.org/data-structures/linked-list/

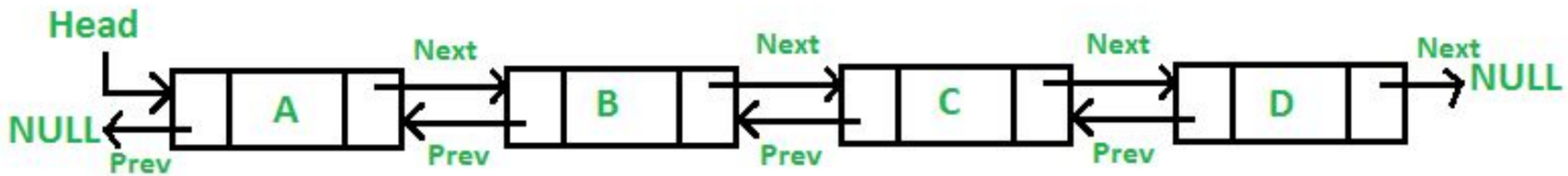
```
class LinkedList {
    Node head; // head / cabeza

    /* Nodo */
    class Node {
        int data;
        Node next;

        // Constructor.
        // next es null.
        Node(int data) { this.data = data; }
    }
}
```

Linked Lists

En un **Doubly Linked Lists**, cada nodo tiene una referencia al **nodo siguiente** y al **nodo anterior**



https://www.geeksforgeeks.org/doubly-linked-list/

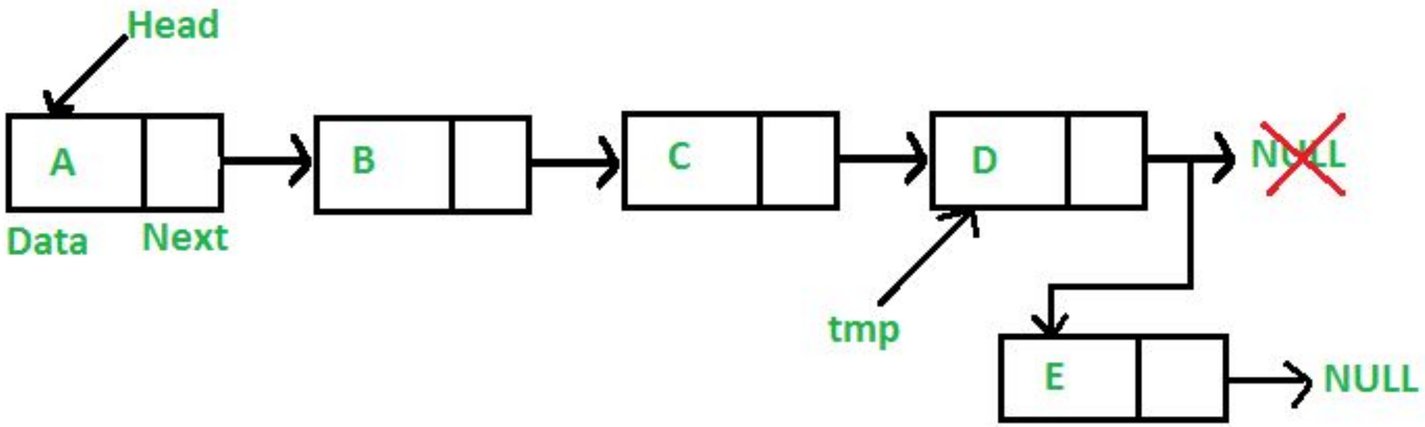
```java
// Doubly Linked List
public class DoublyLinkedList {
    Node head; // head

    /* Node*/
    class Node {
        int data;
        Node prev;
        Node next;

        // Constructor
        // next and prev son null.
        Node(int data) { this.data = data; }
    }
}
```

Ejemplo:

Cómo agregar un nuevo valor al final - *tail*



https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/

```java
public static Node append(Node head, int newData) {
    Node newNode = new Node(newData);

    if (head == null) {
        return newNode;
    }

    Node currentNode = head;
    while (currentNode.next != null)
        currentNode = currentNode.next;

    currentNode.next = newNode;

    return head;
}
```

	Arrays	Linked Lists
Indexado	O(1)	O(n)
Inserción/eliminación al final (Una vez con la referencia al nodo a editar)	O(1)	O(1)
Inserción/eliminación a la mitad (Una vez con la referencia al nodo a editar)	O(n)	O(1)

Estrategias

Linked Lists

Estrategias comunes para problemas de Linked Lists

- Traverse
  - Recorremos todos los nodos de inicio a fin, mientras validamos o buscamos: por ej, cuál es elemento más grande/pequeño dentro de la Linked List?
- The "Runner" Technique / Two pointers
  - Según el problema, podemos usar dos punteros que "avanzan" a diferente velocidad para encontrar la solución.
- Link Manipulation / Modificación de apuntadores
  - Como en el ejemplo anterior, esta estrategia aplica cuando hay que insertar o borrar nodos del Linked Lists
- Recursividad
  - Spoiler - retomaremos el tema más adelante

USOS

Linked Lists

Usos Comunes

- Stacks y Queues
- Operaciones aritméticas de números muy grandes
- Carrete de imágenes
  - Con una doubly linked list podemos representar una foto seleccionada en el carrete, más la foto anterior y la que sigue
- Historial de navegador web
  - Back/Next

Linked Lists

Actividades

- Lee sobre **Linked lists** (o Listas enlazadas)
  - Si dominas el inglés, GeeksForGeeks.org tiene una excelente sección de linked list. Cuidado con ver soluciones antes de intentar resolver el problema por ti mismx.
  - Artículo de wikipedia de Listas Enlazadas
- **Recomendado** practicar e implementar en el lenguaje de tu preferencia.
  - Crea una clase LinkedList que contengan una referencia al nodo cabeza/head.
  - Implementa el ejemplo anterior "append" en tu clase LinkedList como método no estático.
  - Cómo insertar un nuevo nodo en el lugar nth.
  - Cómo eliminar cualquier nodo, ya sea por índice o por valor.
- Resuelve los problemas requeridos de la semana
- **Bonus:** Resuelve los problemas opcionales de la semana

EJERCICIOS

Problemas Requeridos.

D{VZ

DEVZ

Problema #1

Escribe un programa borre elementos duplicados de un Linked List.

Ejemplos

In: 4 → 5 → 9 → 0 → 5 → 1 → 2

Out: 4 → 5 → 9 → 0 → 1 → 2

In: 1 → 2 → 3 → 3 → 2 → 1

Out: 1 → 2 → 3

Problema #2

Tienes dos números representados por dos linked lists donde el valor de cada nodo representa un dígito. Los dígitos están guardados de manera inversa, de manera que las unidades está en el primer nodo (head), las decenas en el segundo nodo, etc.

Escribe un programa que dado dos linked lists, sume los dos números que representan y regrese esa suma representada también en un linked list.

In: **a: 2 → 3 → 1.** **b: 6 → 0 → 3.** **Equivale a 132 + 306**

Out: resultado: **8 → 3 → 4.** Equivale a **438**

In: **a: 9 → 9 → 9.** **b: 1.** **Equivalente a 999 + 1**

Out: resultado: **0 → 0 → 0 → 1.** Equivale a **1000**

DEVZ

Problema #3

Implementa un algoritmo que encuentre nodo k lugares antes del último nodo.

In: **2 → 3 → 1 → 4 → 9 → 10 → 11.** **k = 2**

Out: **9.** Hay dos lugares entre el nodo 9 y el último nodo

In: **1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10.** **k = 9**

Out: **1.** El nodo con valor 1 está en el noveno lugar si tomamos el nodo 10 como referencia.

EJERCICIOS

Problemas Opcionales.

DEVZ

Problema #4

Implementa un algoritmo que borre un nodo que se encuentra dentro de un linked list; éste nodo no es ni el primer ni el último nodo en la secuencia, sólo se sabe que está entre esos dos.

Nota: tu algoritmo no debe de tener acceso al nodo head, sino solo al nodo que se desea borrar.

In: **Nodo "c" del Linked list: a → b → c → d → e → f**

Out: **a → b → d → e → f**

In: **Nodo "y" del Linked list: x → y → z**

Out: **x → z**

Recuerda: tu método tiene como return type ***void***.

Problema #5

Dado un singly linked list, escribe un programa que invierta la dirección de dicha linked list (reverse)

In: **1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10**

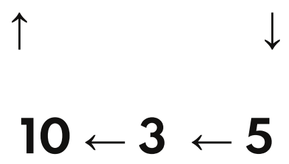
Out: **10 → 9 → 8 → 7 → 6 → 5 → 4 → 3 → 2 → 1**

Puedes hacerlo sin usar ninguna estructura de datos como apoyo?

Problema #6

Dado un singly linked list, escribe un programa que detecte si un linked list tiene una referencia circular

In: 6 → 8 → 0 → 4 → 7 → 2



Out: true. El nodo 10 apunta al nodo 0, lo que crea un loop.

In: 6 → 8 → 0 → 4 → 7 → 2

Out: false

Gracias.

 slack.devz.mx

 @devzcommunity