Estructura de datos y algoritmos

Rodrigo Alvarez

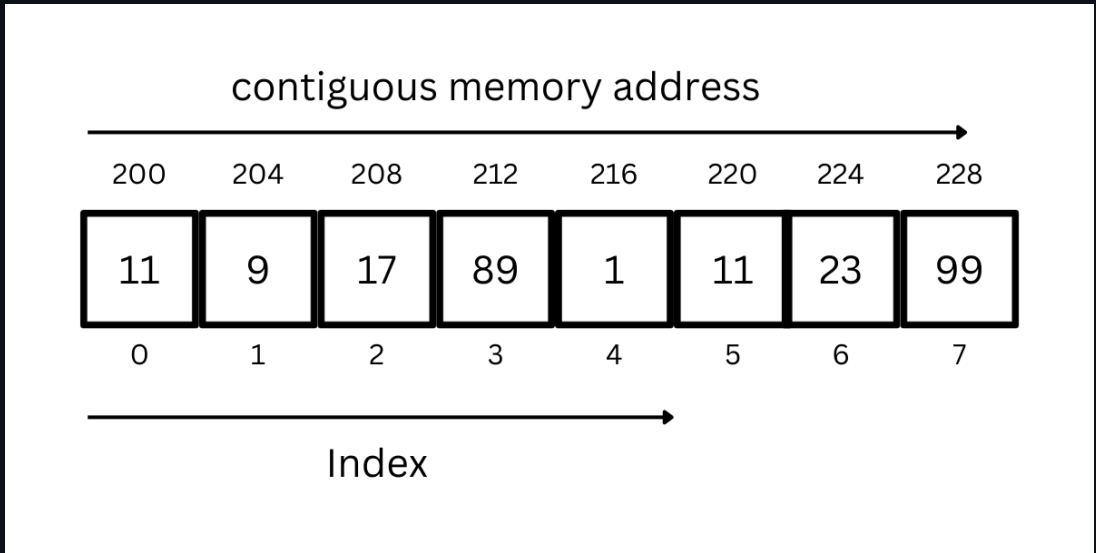
rodrigo.alvarez2@mail.udp.cl

TDA vs Estructuras de datos

- **Tipo de dato abstracto** (TDA)
  - Dice qué hace, no cómo lo hace. (especificación)
  - Que data se puede guardar
  - Qué operaciones se pueden hacer / Qué es lo que significan.
- **Estructura de datos**
  - Dice cómo se hace. (implementación)
  - Cómo se guardan los datos
  - **Cómo se hacen** las operaciones (algoritmos).

Arrays

- Colección de elementos ordenados.
  - Cada elemento es una unidad básica de información.
  - Posición: Lugar que ocupa un elemento en el array.
- Tamaño fijo.



Listas

- Colección de elementos ordenados.
  - Cada elemento es una unidad básica de información.
  - Posición: Lugar que ocupa un elemento en la lista.
- Tamaño es dinámico.

```
interface List {
    int get_at(int i); // obtener elemento en posición i
    void insert_at(int i, int e); // agregar elemento en posición i
    int delete_at(int i); // remover elemento en posición i
    ...
    void insert_at_end(int e); // agregar elemento al final
    void insert_at_start(int e); // agregar elemento al inicio
    int delete_at_end(); // remover elemento al final
    int delete_at_start(); // remover elemento al inicio
}
```

Listas

Usando generics:

```
interface List<T> {
    T get_at(int i); // obtener elemento en posición i
    void insert_at(int i, T e); // agregar elemento en posición i
    T delete_at(int i); // remover elemento en posición i
    ...
    void insert_at_end(T e); // agregar elemento al final
    void insert_at_start(T e); // agregar elemento al inicio
    T delete_at_end(); // remover elemento al final
    T delete_at_start(); // remover elemento al inicio
}
```

Arreglos dinámicos

Es una estructura de almacenamiento de datos que cambia de tamaño dinámicamente conforme los elementos se agregan o se eliminan.

Listas enlazadas

Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior

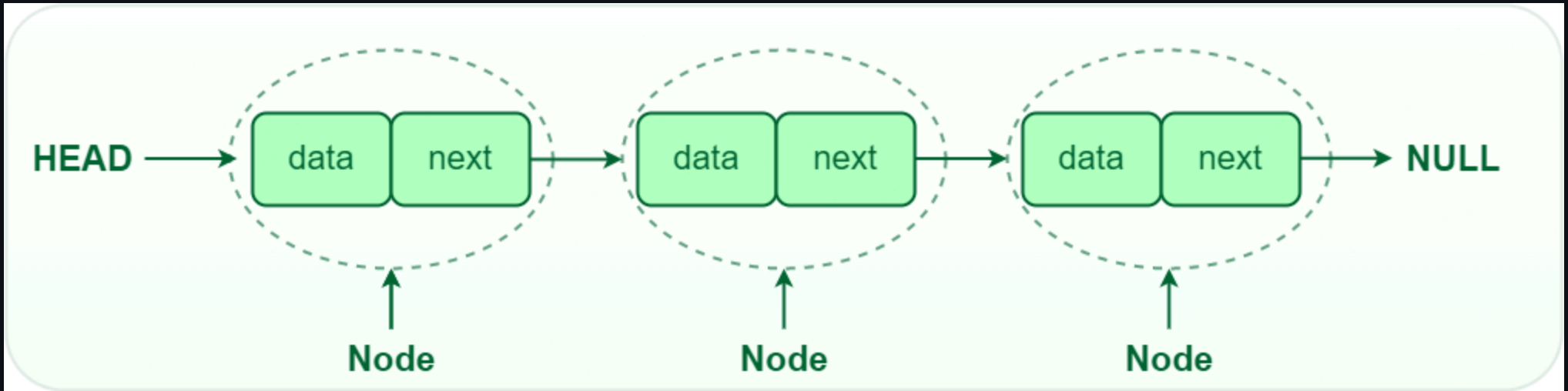
Nodo

- Estructura básica utilizada en la construcción de la mayoría de las estructuras de datos
- Está compuesto por los datos que almacena y una lista de referencias a otros nodos
  - En el caso de las listas (básicas), pilas y colas solo contiene una referencia
  - En el caso de los árboles contiene dos o más referencias

```
class Node {
  int data;
  Node next;
  public Node(int data) {
    this.data = data;
    this.next = null;
  }
  public int getData() { return data; }
  public void setData(int data) { this.data = data; }
  public Node getNext() { return next; }
  public void setNext(Node next) { this.next = next; }
}
```

Lista enlazada

- Lista de nodos enlazados
- El último nodo apunta a `null`
- El primer nodo se llama `head`



Lista enlazada

Lista enlazada

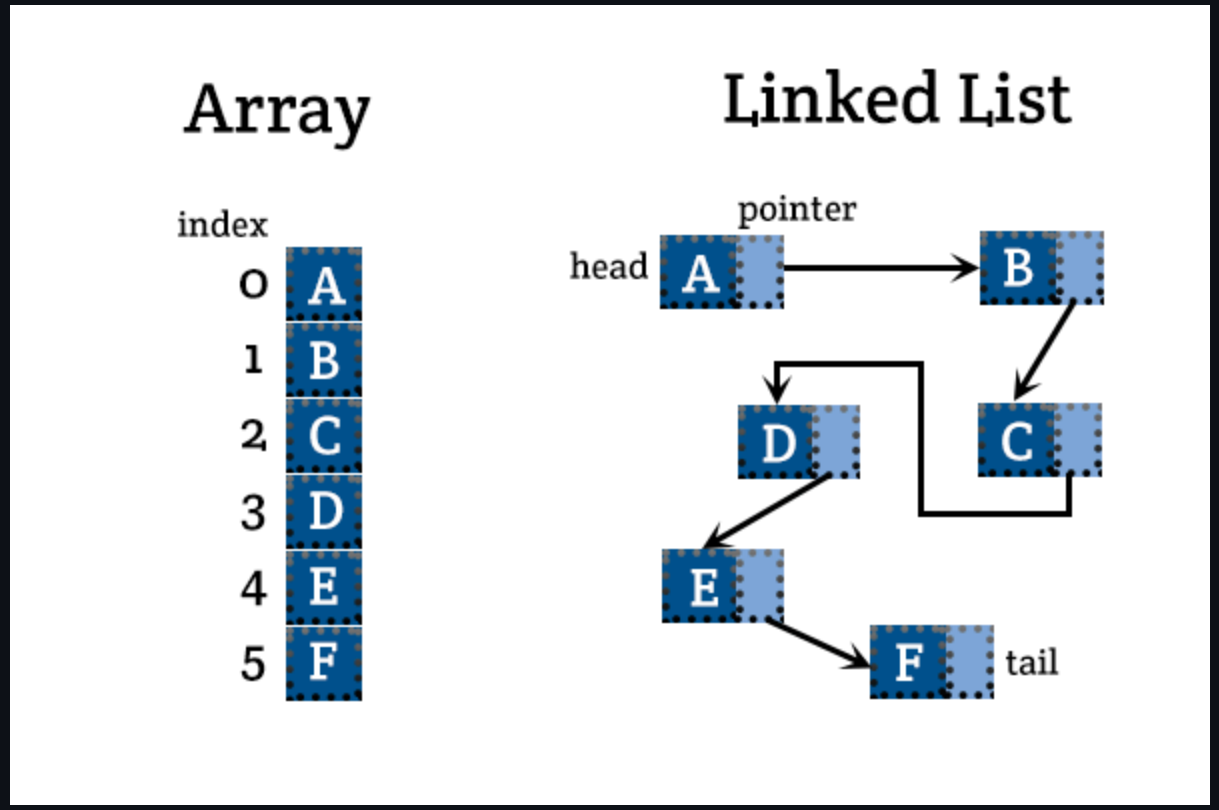
```
class LinkedList {
  Node head;
  public LinkedList() {
    this.head = null;
  }
  public int get_at(int i) {
    Node current = head;
    for (int j = 0; j < i; j++) {
      current = current.next;
    }
    return current.data;
  }
  ...
}
```

```
class LinkedList {
  ...
    public void insert_at(int i, int e) {
    Node newNode = new Node(e);
    if (i == 0) {
      newNode.next = head;
      head = newNode;
    } else {
      Node current = head;
      for (int j = 0; j < i - 1; j++) {
        current = current.next;
      }
      newNode.next = current.next;
      current.next = newNode;
    }
  }
  ...
}
```

Lista enlazada

```
class LinkedList {
  ...
  public int delete_at(int i) {
    if (i == 0) {
      int data = head.data;
      head = head.next;
      return data;
    } else {
      Node current = head;
      for (int j = 0; j < i - 1; j++) {
        current = current.next;
      }
      int data = current.next.data;
      current.next = current.next.next;
      return data;
    }
  }
}
```

Array vs lista enlazada



Lista enlazada con tail

- Lista enlazada con referencia al último nodo
- El último nodo se llama `tail`

```
class LinkedListWithTail {
  Node head;
  Node tail;
  public LinkedList() {
    this.head = null;
    this.tail = null;
  }
  public void insert_at_end(int e) {
    Node newNode = new Node(e);
    if (head == null) {
      head = newNode;
      tail = newNode;
    } else {
      tail.next = newNode;
      tail = newNode;
    }
  }
}
```

Complejidad

	get_at	insert_at	delete_at	insert_at_end	insert_at_start	delete_at_enc
Array	O(1)	O(n)	O(n)	O(1)	O(n)	O(n)
LinkedList	O(n)	O(n)	O(n)	O(n)	O(1)	O(n)
LinkedList + tail	O(n)	O(n)	O(n)	O(1)	O(1)	O(n)

LinkedList en la stdlib de java

- `add(int index, E e)` Inserta un elemento en una determinada posición
- `addFirst(E e)` Agrega un elemento al principio de la lista
- `addLast(E e)` Agrega un elemento al final de la lista
- `remove(int index)` Elimina un elemento en una determinada posición
- `removeFirst()` Elimina el elemento al principio de la lista
- `removeLast()` Elimina el elemento al final de la lista
- `get(int index)` Obtiene el elemento en una determinada posición
- `getFirst()` Obtiene el elemento al principio de la lista
- `getLast()` Obtiene el elemento al final de la lista

42r347rgd

JAVA

RUN

```
1
```

STDIN

Input for the program (Optional)

Output:

```
Main.java:13: error: variable cars is already define
      LinkedList<String> cars = new LinkedList<Strin
                         ^
1 error
error: compilation failed
```

- Doubly linked list
- Doubly linked list complexity analyisis