Video 14 (1)

Conceptos Principales

1. a
2. b
3. c

Notas

Listas Dobles, insertar al final

Conceptos clave

- Una **lista doble** es una variación de la lista simple.
- Cada **nodo** tiene:
  - `valor`
  - `siguiente` (puntero al nodo que sigue)
  - `anterior` (puntero al nodo que lo precede).
- Ventaja: permiten **recorridos en ambas direcciones** → útiles en algoritmos que necesitan moverse dentro de la lista sin ir desde el inicio.

Estructura básica

- Se define un `struct ListaDoble` con un **puntero al inicio**.
- Cada `Nodo` de la lista enlaza hacia adelante y hacia atrás.
- Caso inicial: si la lista está vacía (`inicio == NULL`), el primer nodo se convierte en el inicio y ambos punteros (`siguiente` y `anterior`) quedan en `NULL`.

Algoritmo de inserción al final

1. **Crear nodo nuevo** con el valor deseado, punteros `siguiente` y `anterior` en `NULL`.
2. Si la lista está vacía → el nuevo nodo pasa a ser el inicio.
3. Si no está vacía:
   - Se usa un puntero auxiliar (`actual`) para recorrer desde el inicio.
   - Se avanza mientras `actual->siguiente != NULL`.
   - Al llegar al último nodo, se ajustan los punteros:
     - `actual->siguiente = nuevo;`
     - `nuevo->anterior = actual;`

Ejemplo visto en clase

- Insertar en orden: **18, 27, 3, 19**.
- Primer nodo (`18`) → inicio de la lista.
- Insertar `27` → se conecta al final (`18 <-> 27`).
- Insertar `3` → se recorre hasta el final y se conecta (`18 <-> 27 <-> 3`).
- Insertar `19` → igual proceso (`18 <-> 27 <-> 3 <-> 19`).

Observaciones

- Mucho cuidado con el **orden de asignación de punteros**:
  - Primero conectar `siguiente` del último nodo.
  - Luego conectar `anterior` del nuevo nodo.
- La verdadera diferencia con listas simples se nota más en **inserción ordenada** y **eliminación**, que requieren más manejo de punteros.