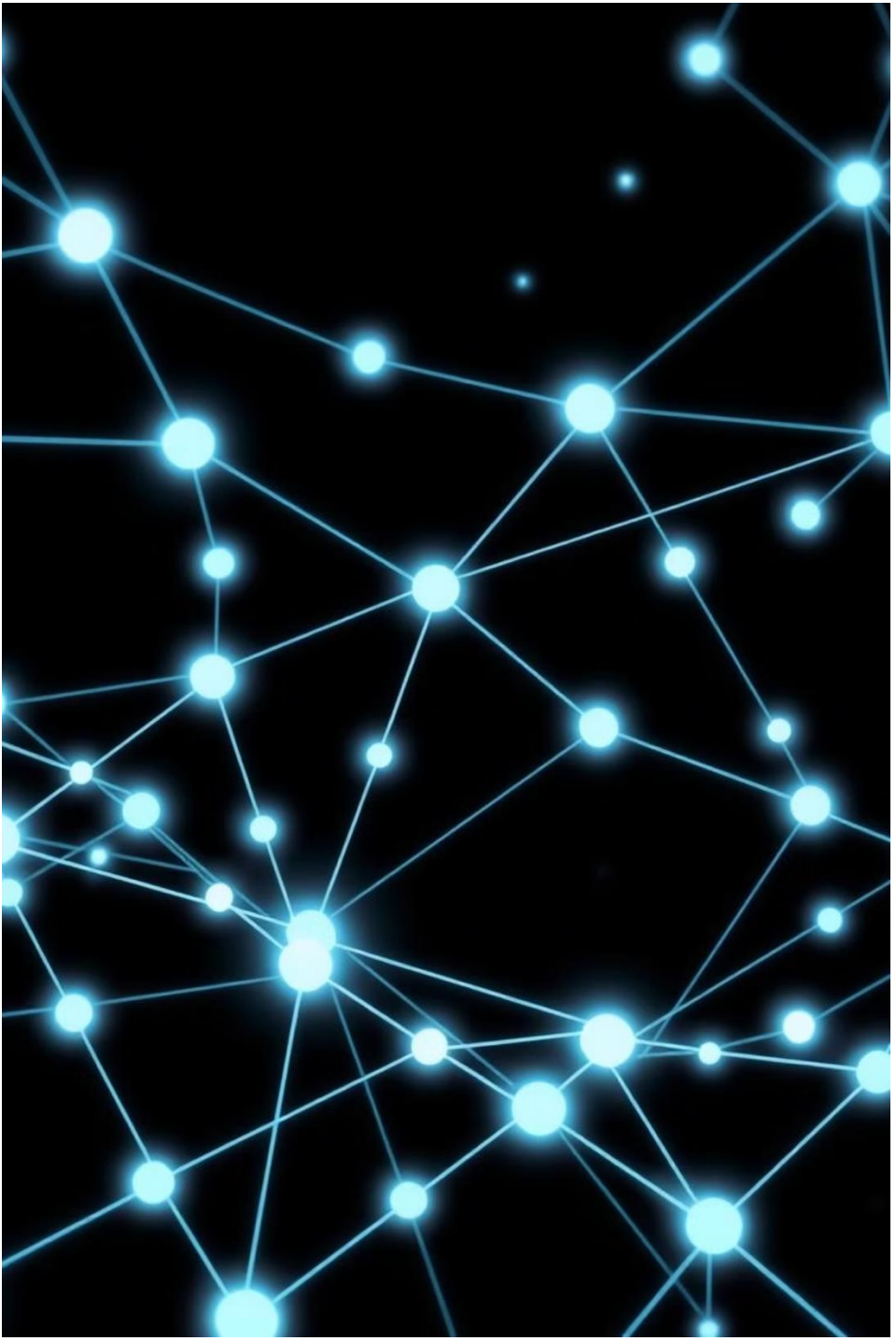


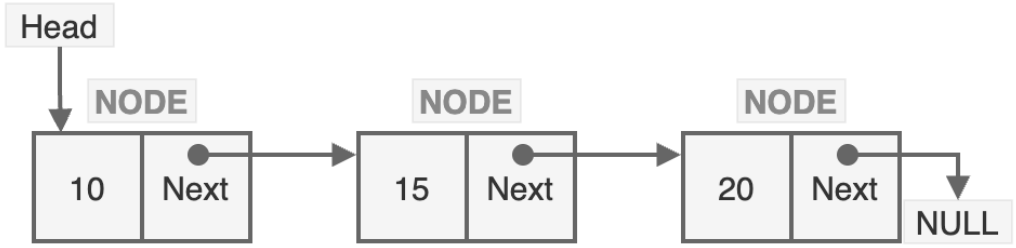
Data Structures

Lecture 4: Linked Lists

Contents

- Concept of Linked List
- Advantages and Disadvanatages of Linked List
- Array Vs Linked Lists
- Operations in Linked List
- Types of Linked List
- Algorithms & Pseudocode
- Implementation

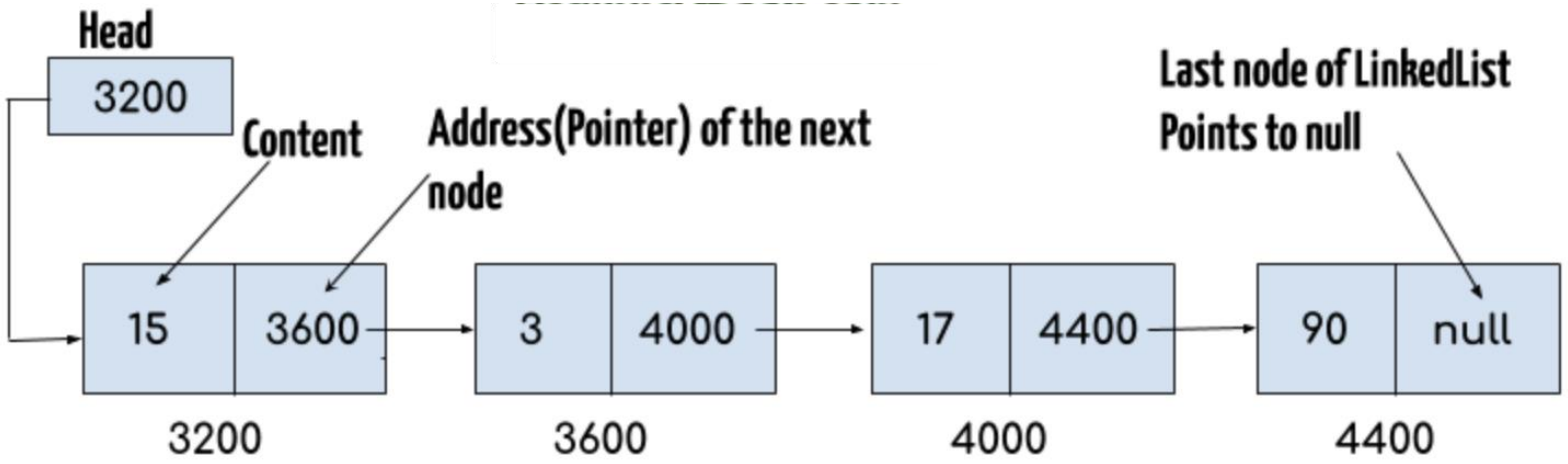
Linked Lists: Flexible Chains of Nodes



- A linked list is a collection of linear items. In contrast to the array, a linked list stores the elements in **non-contiguous memory** locations.
- Linked lists consist of nodes, each containing data and a pointer (or reference) to the next node in the sequence.
- Linked List is best used for storing dynamic data that requires frequent insertion and deletion operations.
- Supported operations are ***Insertion, Deletion, and Search***.

Linked List

Node→ Data + Pointer



Operations in Linked List

- ***Traversal/ Display***
- ***Searching (Sorted or unsorted List)***
- ***Insertion***
- ***Deletion***

Insertion

- The insert operation adds a new element to the linked list. The following tasks are performed while adding the new element:
  - (a) Memory space is reserved for the new node.
  - (b) The element is stored in the INFO part of the new node.
  - (c) The new node is connected to the existing nodes in the list.
- ***Three*** Possible scenarios for adding new node:
  - (a) Inserting the new element at the beginning of the list
  - (b) Inserting the new element at the end of the list
  - (c) Inserting the new element somewhere at the middle of the list

Inserting at the middle of the List

- ***Search operation*** is required to be performed to identify the point of insertion.

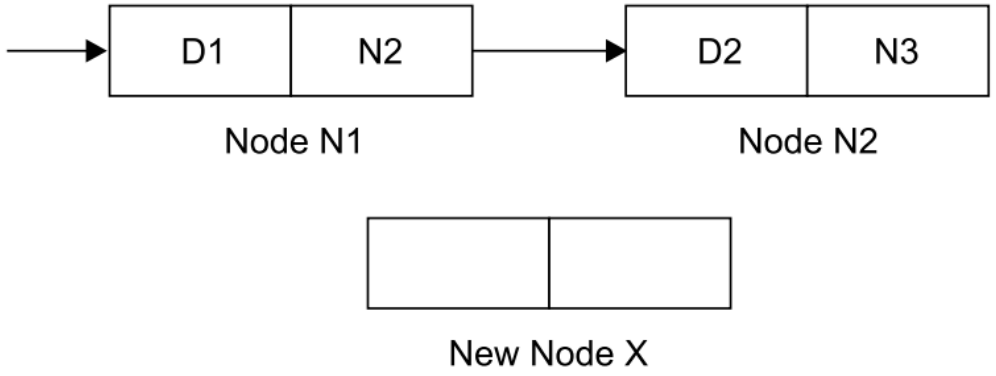


Fig 5.2 (a) *Creating a new element*

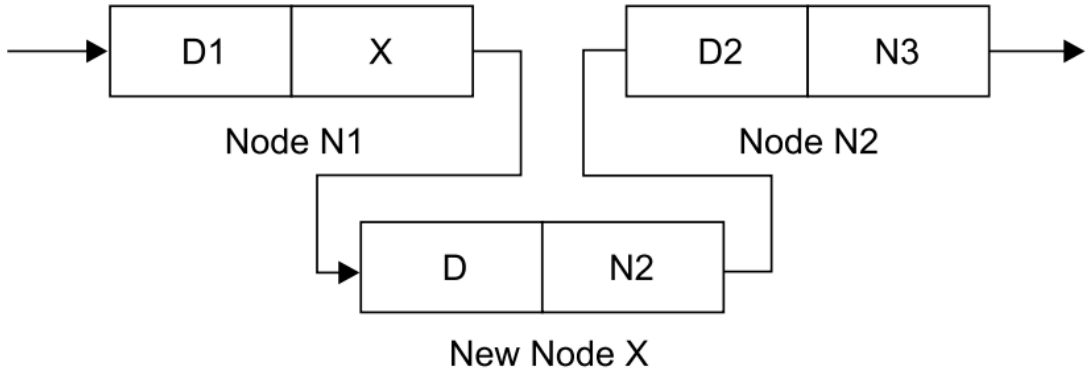


Fig 5.2 (b) *Inserting the newly created element*

Deletion

- The delete operation *removes an existing* element from the linked list. The following tasks are performed while deleting an existing element:
  - (a) The location of element is identified.
  - (b) The element value is retrieved. In some cases, the element value is simply ignored.
  - (c) The link pointer of the preceding node is reset.
- ***Three Possible*** Scenarios of Deleting a Node.
  - (a) Deleting an element from the beginning of the list.
  - (b) Deleting an element from the end of the list.
  - (c) Deleting an element somewhere from the middle of the list.

Deleting at any point of the Linked List

- Search operation is required to be performed for locating that element.

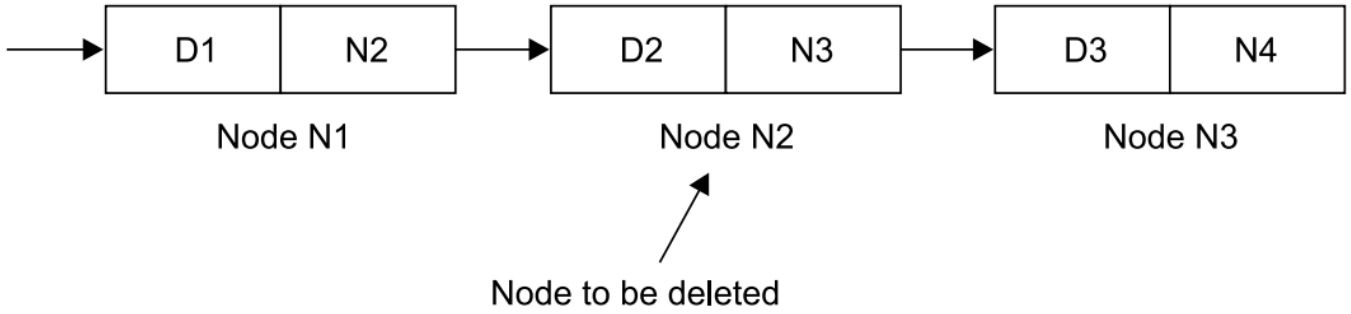


Fig 5.3 (a) *Identifying the node to be deleted*

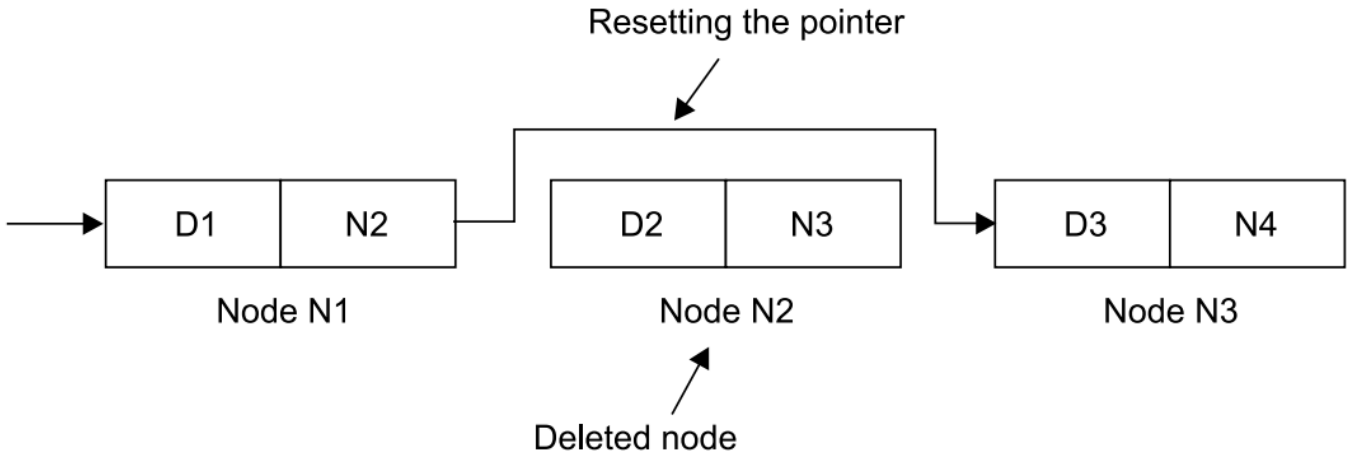


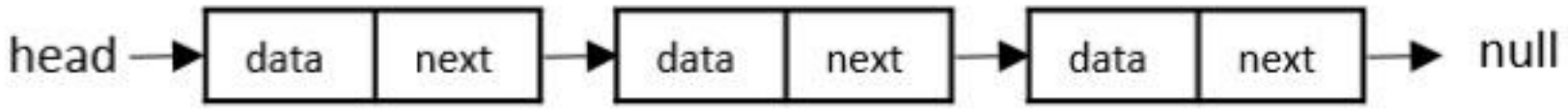
Fig 5.3 (b) *Deleting the node*

Linked List

- **Types**
  - Singly Linked List
  - Doubly Linked List
  - Circular Linked List
  - Doubly Circular Linked List
- **Applications**
  - Music playlist navigation
  - Web browser back/forward history
  - Undo functionality in software
  - Operating system's memory management

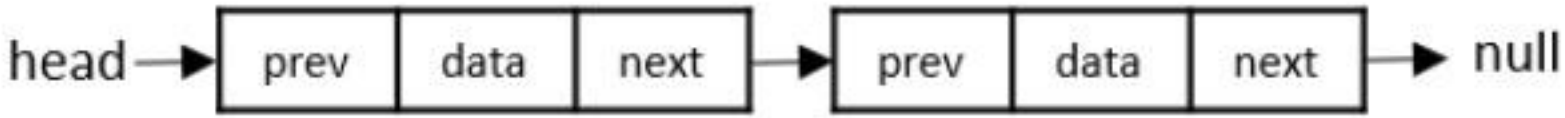
Singly Linked List

- Singly Linked List contains ***two buckets/ Fields*** in each node.
- One bucket holds the data and other holds the address of next node.
- Traversal can be done in one direction (*Forward → Head node to tail node*).



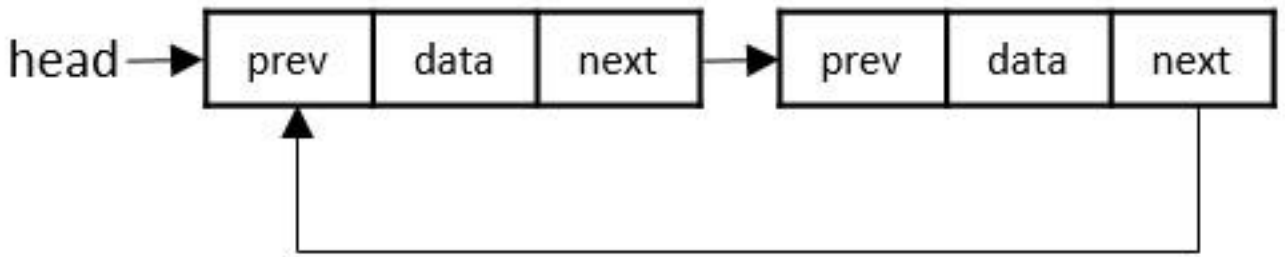
Doubly Linked List

- Doubly Linked Lists contain ***three buckets*** in one node.
- One bucket holds the ***data*** and the other buckets hold the addresses of the ***previous*** and ***next*** nodes in the list.
- Traversal can be done in both directions (*Forward and backward*).



Circular Linked List

- Circular linked lists can exist in both singly linked list and doubly linked list.
- Since the last node and the first node of the circular linked list are connected, the traversal in this linked list will go on forever until it is broken.



References

- **Chapter 5:**
  - **Data Structures using C** by E. Balagurusamy
- **Chapter 10 (Section 10.2) by Cormen**

Thank You