Data structures

Doubly Linked List

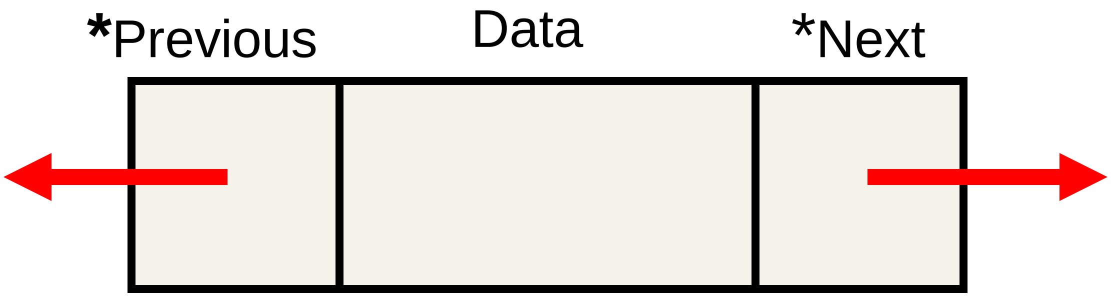
Lecture No. 8

Introduction

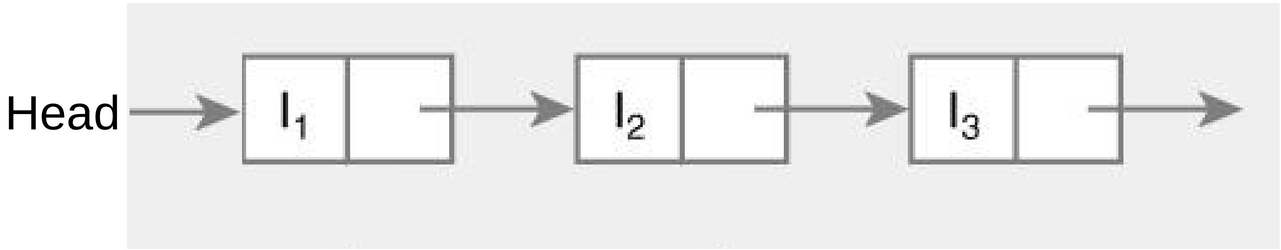
- The singly linked list contains only one pointer field i.e. every node holds an address of next node.
- The singly linked list is uni-directional i.e. we can only move from one node to its successor.
- This limitation can be overcome by **Doubly linked list.**

Doubly Linked List

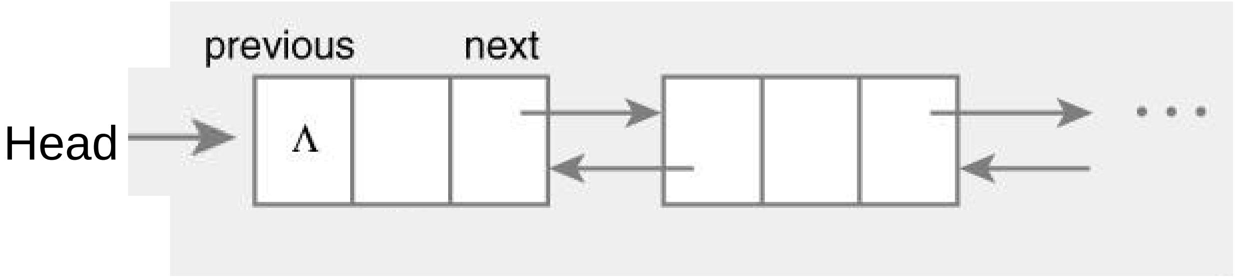
- In Doubly linked list, each node has two pointers.
- One pointer to its successor (NULL if there is none) and one pointer to its predecessor (NULL if there is none).
- These pointers enable bi-directional traversing.



A Singly Linked List



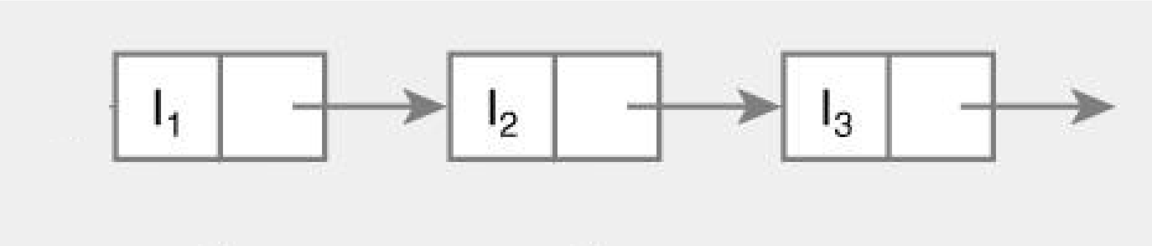
A Doubly Linked List



Comparison of Linked Lists

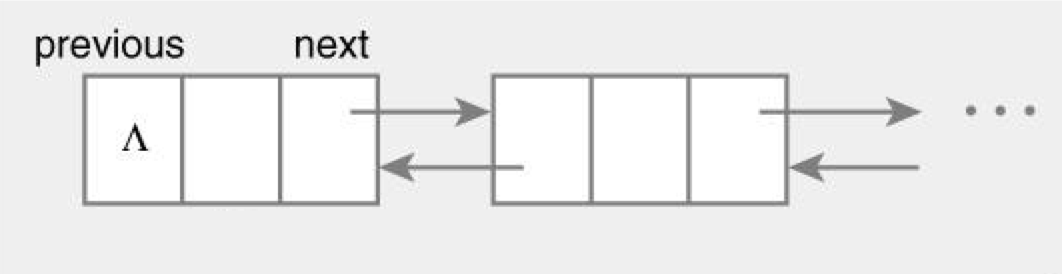
- Linked list

```
struct Node {
   int data;
   Node* next;
};
```



- Doubly linked list

```
struct Node {
   Node *previous;
   int data;
   Node *next;
};
```

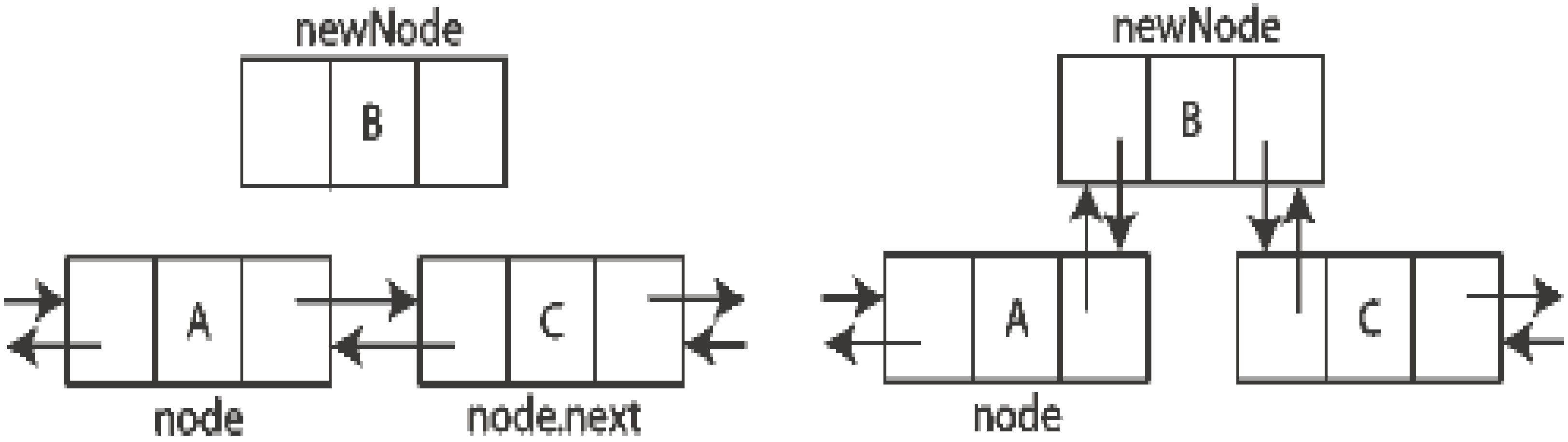


Insertion

- In insertion process, element can be inserted in three different places
    - At the beginning of the list
    - At the end of the list
    - At the specified position.

- To insert a node in doubly linked list, you must update pointers in both predecessor and successor nodes.

Insertion

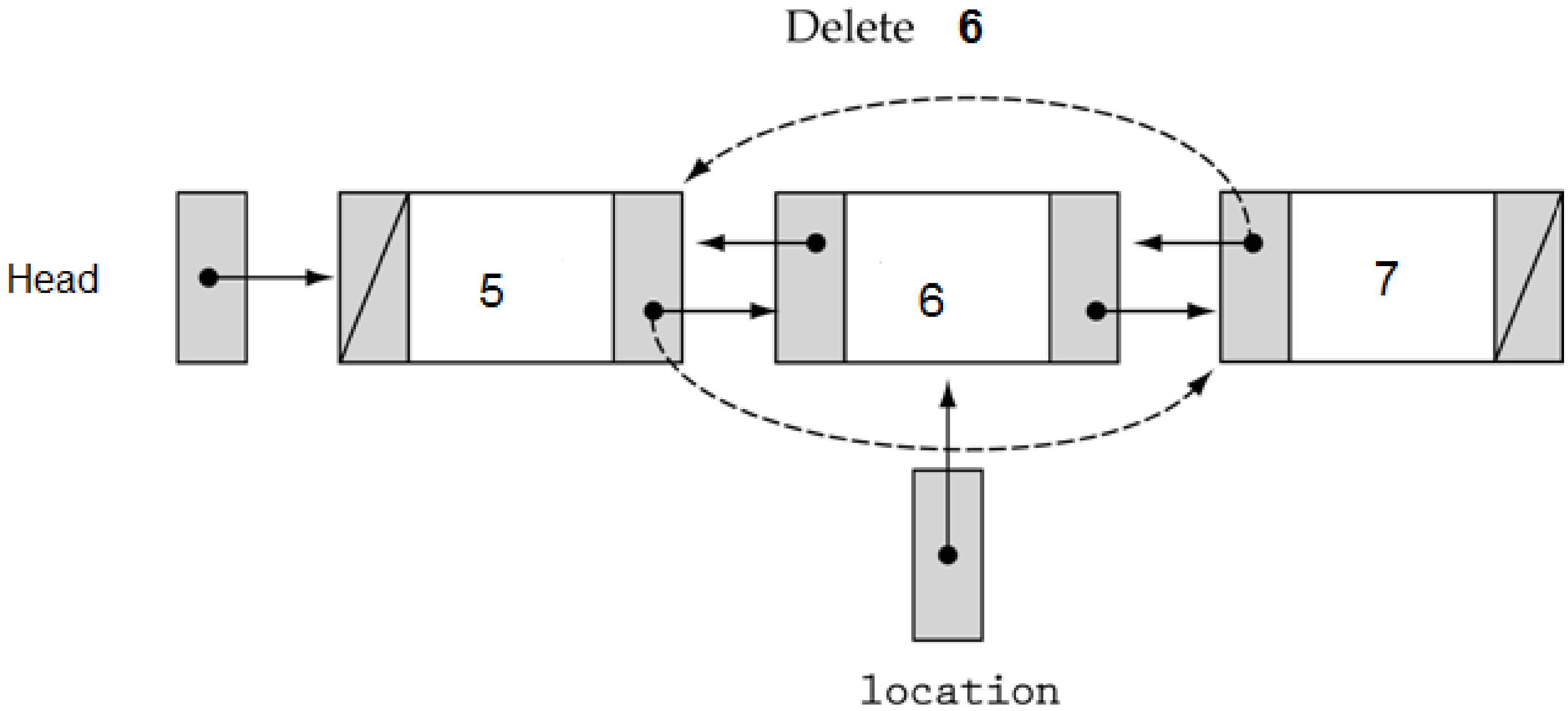


Deletion

- In deletion process, element can be deleted from three different places
  - From the beginning of the list
  - From the end of the list
  - From the specified position in the list.

- When the node is deleted, the memory allocated to that node is released and the previous and next nodes of that node are linked

Deletion



Advantages of Doubly Linked List

- The doubly linked list is bi-directional, i.e. it can be traversed in both backward and forward direction.
- The operations such as insertion, deletion and searching can be done from both ends.
- Predecessor and successor of any element can be searched quickly.

Disadvantages

- It consume more memory space.
- There is a large pointer adjustment during insertion and deletion of element.
- It consumes more time for few basic list operations.