Data Structures and Algorithms

Doubly Linked List

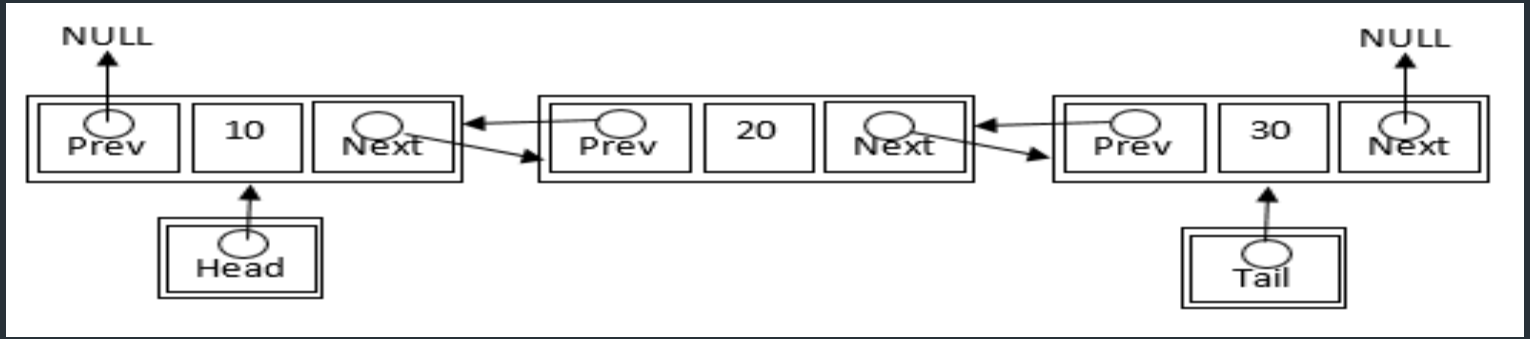
Contents

- Introduction
- Representation
- Basic Operations & Procedure
- Advantages & Disadvantages

Introduction

- Doubly Linked List (DLL) is a data structure belonging to Linked List family
- Here, navigation is possible in both forward and backward ways
- It is a modification to Singly Linked List (SLL) where only forward navigation is possible

Representation

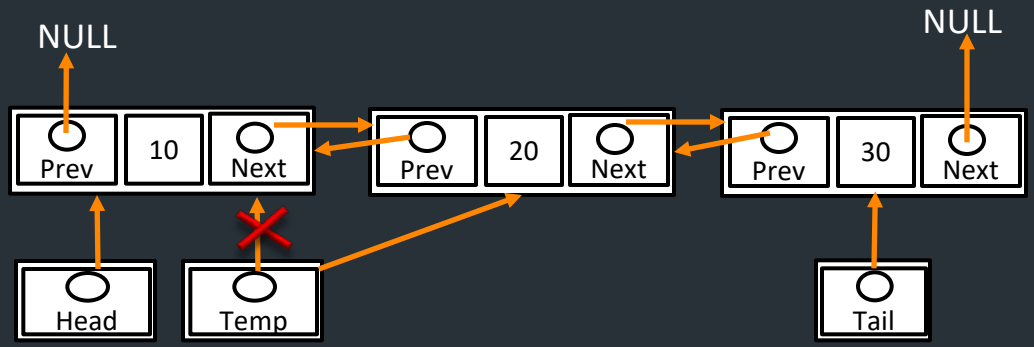


- Each node must contain at least one data field
- Each node must contain two information fields (Next and Prev)
- Next - stores the address of the next node
- Prev - stores the address of the previous node
- Head points to first and tail points to last node
- Head's Prev and Tail's Next is assigned to NULL

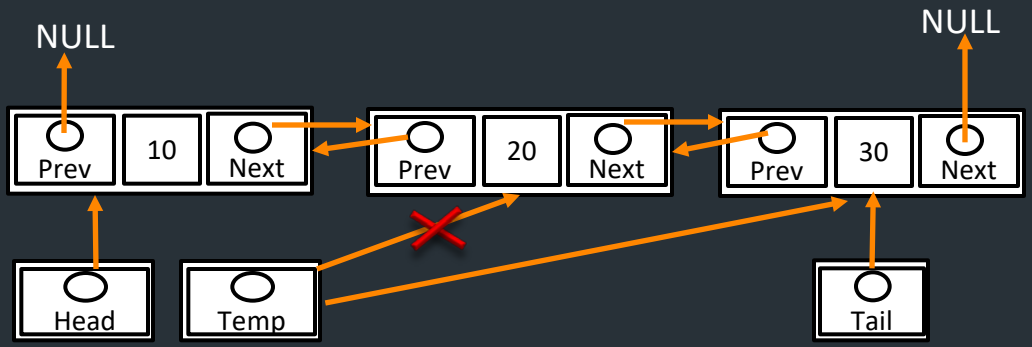
Basic Operations

- Display
- Traversal in Reverse direction
- Search

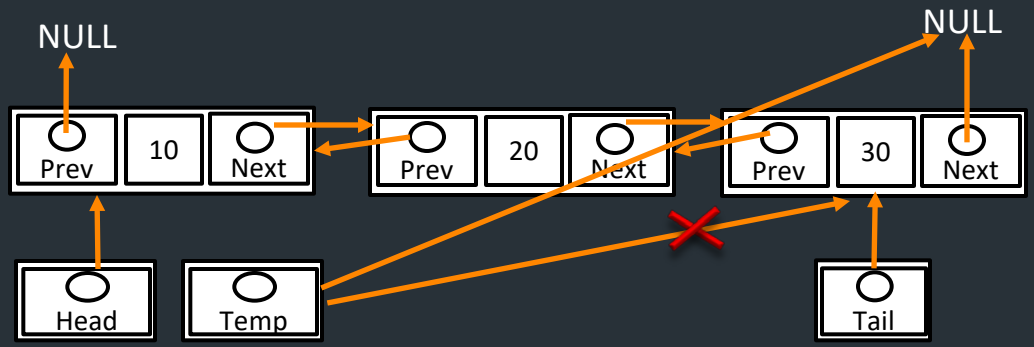
Display the elements in the list



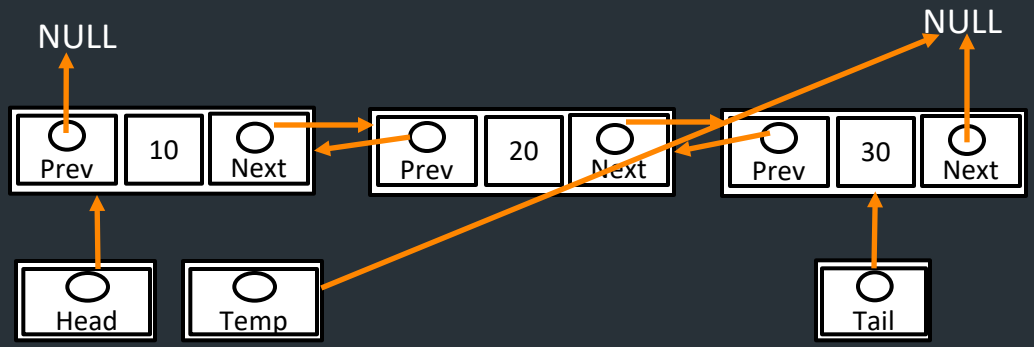
Display the elements in the list



Display the elements in the list



Display the elements in the list



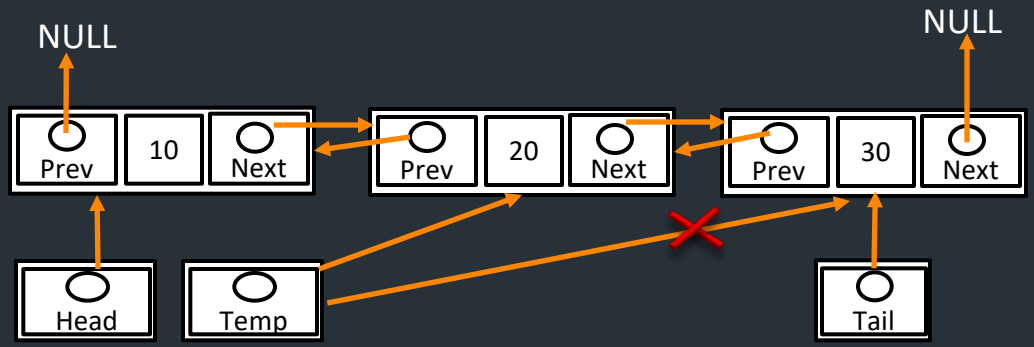
Display the elements in the list

- Temp <= Head
- While (Temp != NULL)

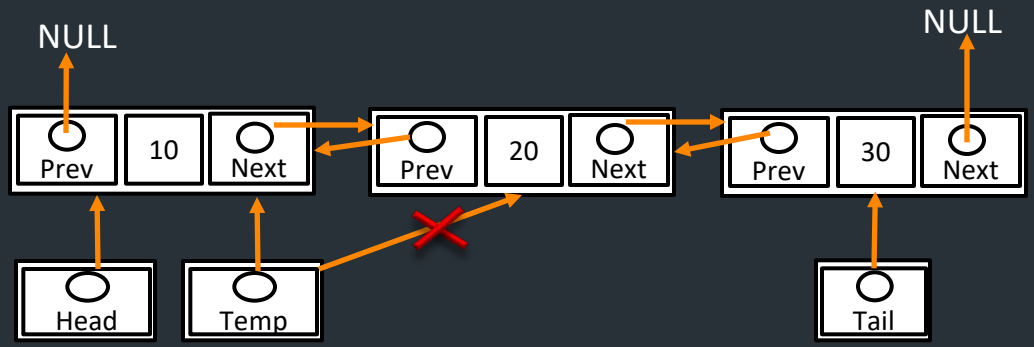
  print Temp(data)

  Temp <= Temp(Next)

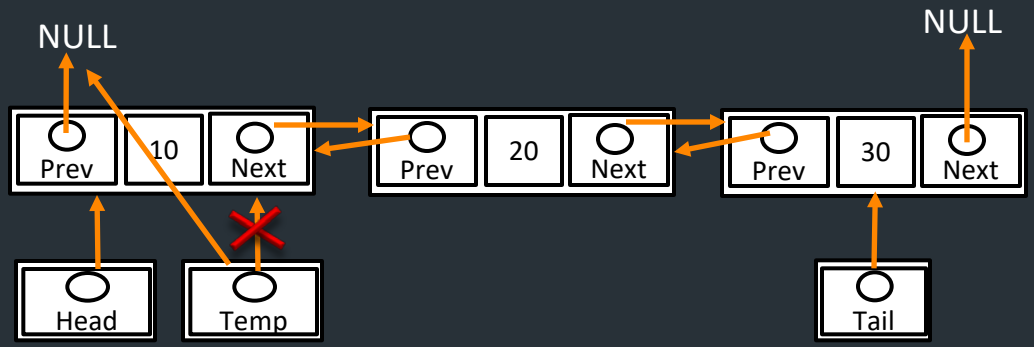
Display the elements in the list in reverser order



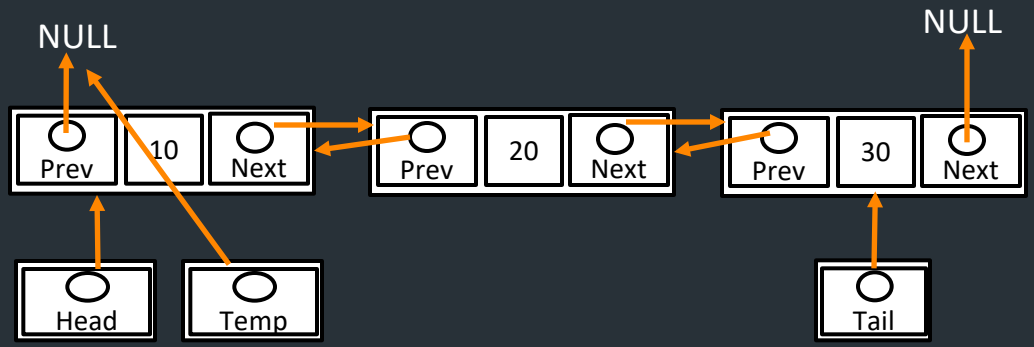
Display the elements in the list in reverser order



Display the elements in the list in reverser order



Display the elements in the list in reverser order



Display the elements in the list

- Temp <= Tail
- While (Temp != NULL)

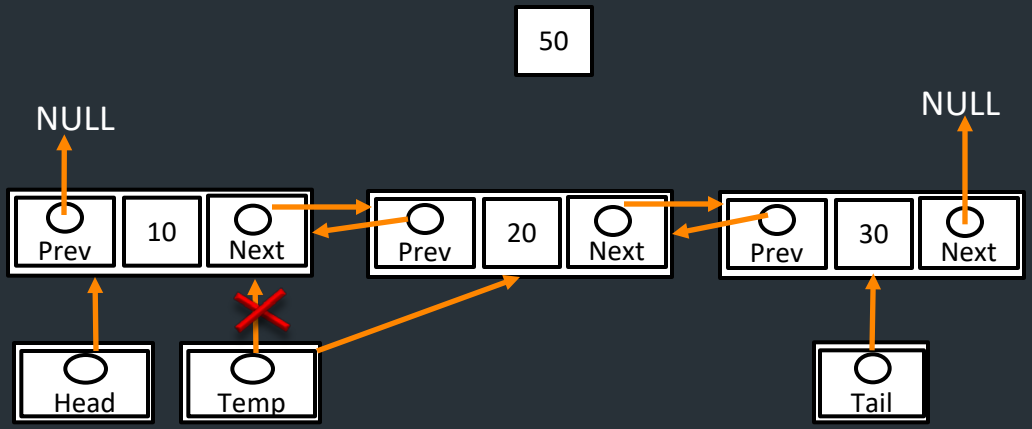
  print Temp(data)

  Temp <= Temp(Prev)

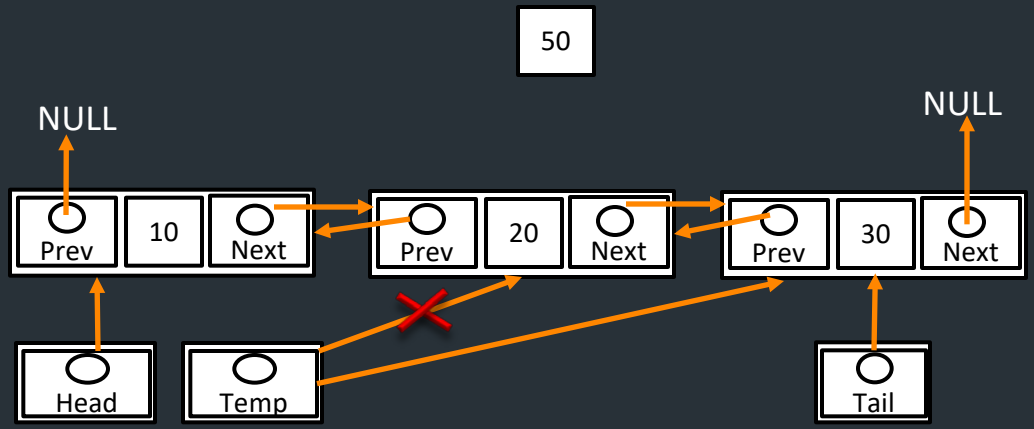
Display

- Time Complexity: O(n)
- Space Complexity: O(n)

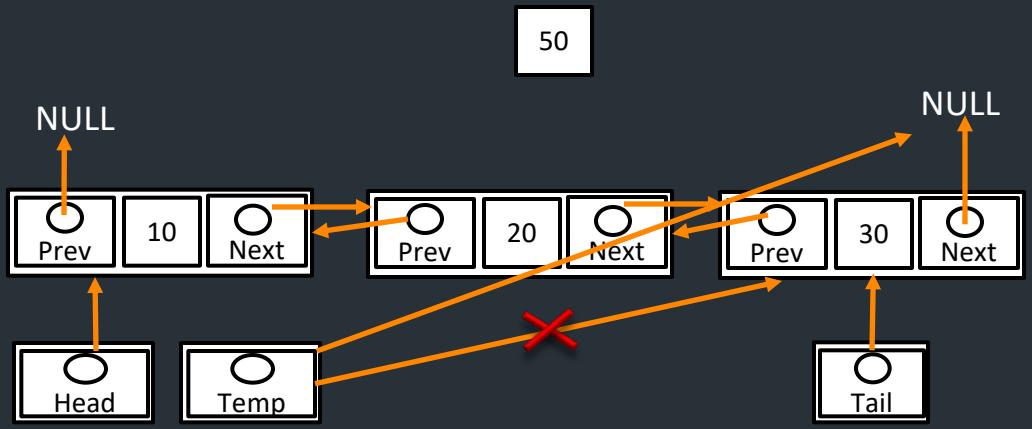
Search an item



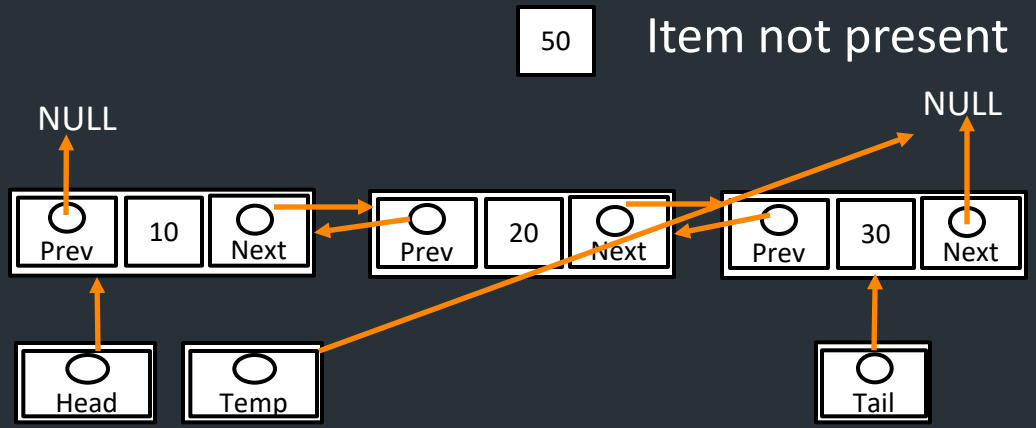
Search an item



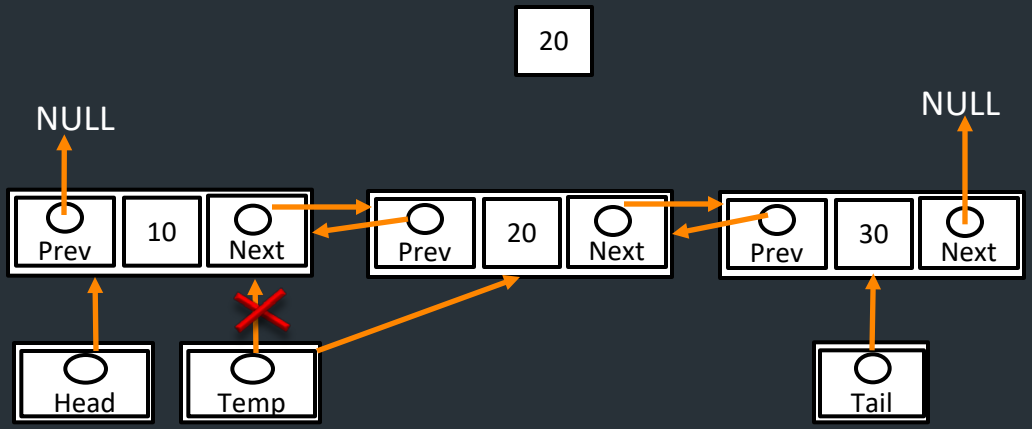
Search an item



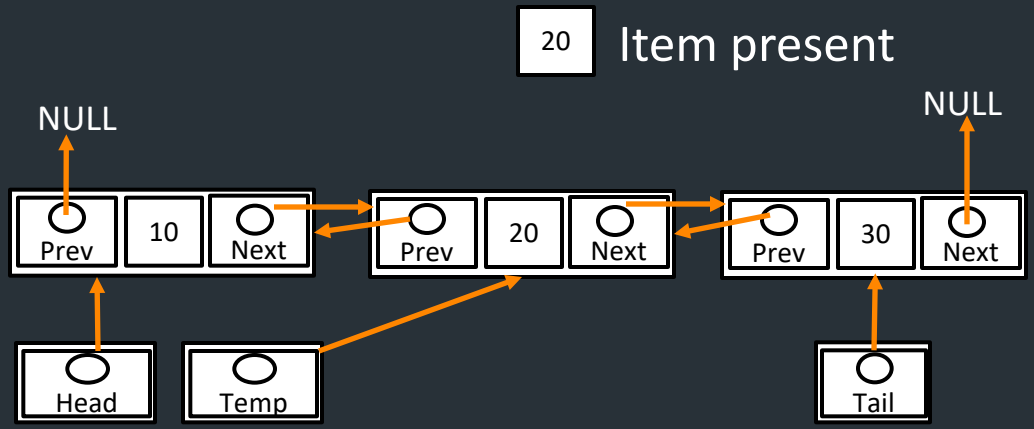
Search an item



Search an item



Search an item



Search an item

- Scan (item)
- Temp <= Head
- While (Temp != NULL)

  if Temp(data) == item

  print “Item Present”

  break

  Temp <= Temp(Next)
- If (Temp == NULL)

  print “Item not present”

Search an item

- Time Complexity: O(n)
- Space Complexity: O(n)

Thank You