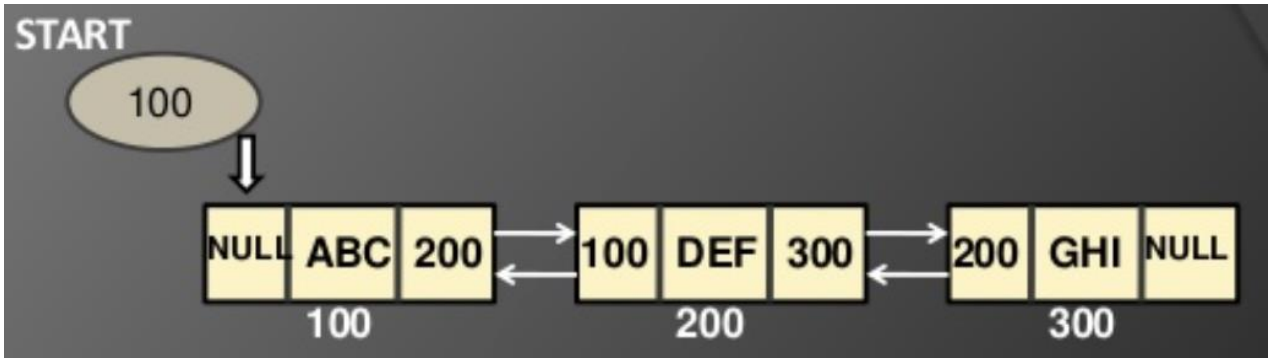
IBA

IBA FCS
Faculty of Computer Science

CSE 247
Data Structure

Doubly linked list

- Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List



- Data could be
  - one single information or more than one information or an object of user defined data type.

Doubly Linked List (DLL)

- Why Doubly linked list ?
  - In singly linked list we cannot traverse back to the previous node without an extra pointer. For example cannot get previous node.
  - In doublylist there is a link through which we can go forward as well as back to the previous node.

Operations on DDL

- Operations On Doubly Link List are
  - Insertion
    - At First
    - At Last
    - At Desired
  - Deletion
    - At First
    - At Last
    - At Desired
  - Traversing
    - Lookup

Basic operations on DLL

Nested referencing:

- temp.next.next.next;
- temp.next.data; and so on

Traverse a list

- a linked list that require the link to be traversed
  - Search the list for an item
  - Insert an item in the list
  - Delete an item from the list

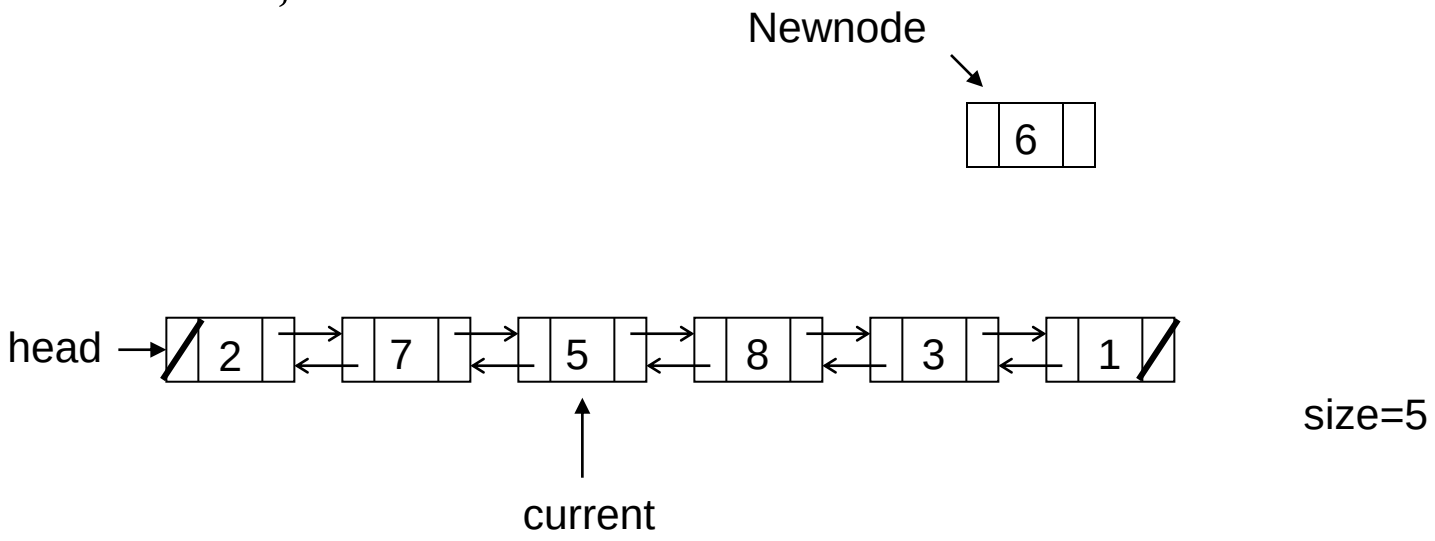
```
current = head;
while (current != null)
{
    current = current.next;
}
```

- You cannot use `head` to traverse the list
  - You would lose the nodes of the list
  - Use another reference variable of the same type as `head`: `current`

Insertion in a doubly linked list

- Insertion as first node

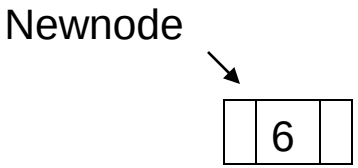
```
Node Newnode=new Node(V);
Newnode.next=Head;
Head.prev=Newnode;
Head=Newnode;
```

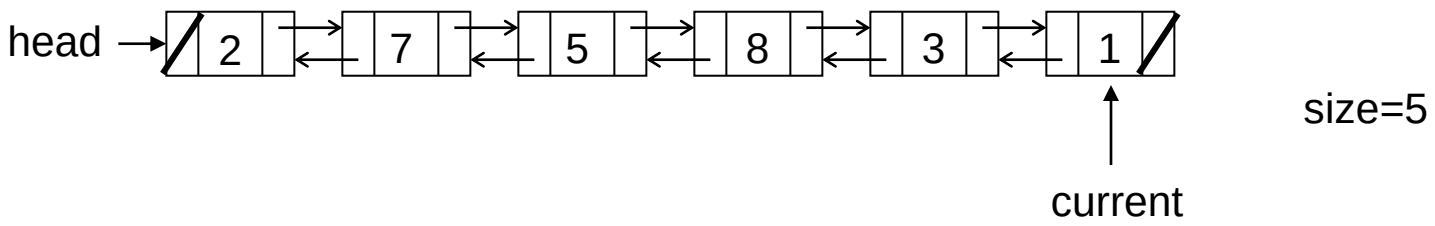


Insertion in a doubly linked list

- Insertion as last node when tail is not define

```
Node Newnode=new Node(V);
Node current=head;
while(current.next!=null) {current=current.next;}
current.next=Newnode;
Newnode.prev=current;
```

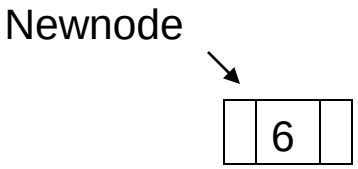


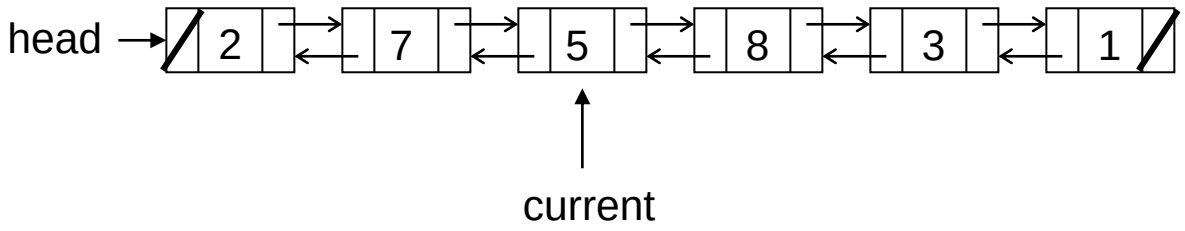


Insert in a middle of a list after curr

```
Node Newnode=new Node(V);
 Newnode.prev=curr;
 Newnode.next=curr.next;

Curr.next.prev=Newnode;
Curr.next=Newnode;
```





Find data in a doubly linked list

```
Public Boolean find(int d){
Node current=head;
while(current!=null && d!=current.data ) {
   current=current.next;
}
If(current==null){
return false;
}
else {
return true;
}
}
```

Deletion from a doubly linked list

- Deletion in middle using curr:

  curr.prev.next=curr.next;

  curr.next.prev=curr.prev;

- Deletion of a first node

  Head.next.prev=null;

  Head=Head.next;

- Deletion of a last node

  Current.prev.next=null;

DLL Head and Tail pointers

- Head pointer
  - Do not change head pointer.
- Tail pointer
  - Insertion at last can perform in constant time by taking tail pointer

Other operations on DLL

- **Find length of a DLL.**

```
public int length(){
    int count=0;
   while(current!=null && d!=current.data ) {
       current=current.next;
       count++;
   }
     return count;
}
```

- **Is the list is empty**

```
public boolean isEmptyList(){
     return (first == null);
}
```

Linked list

- Advantages of linked lists

  (1) Dynamic memory allocation

  (2) Efficient insertion-deletion (for sorted lists)

- Can we implement a linked list without dynamic memory allocation ?

Doubly linked list (DLL)

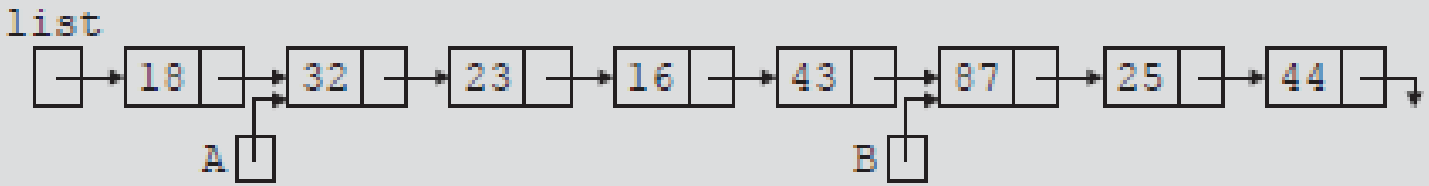
- Advantages of a doubly linked list
  - can be iterated in both direction
  - insert/deletion from double-linked lists at an arbitrary node faster as compare to singly list because you do not have to scan again from the beginning to find the previous node.

- Disadvantages of doubly linked list
  - costs more memory and more operations to insert/delete items as compare to singly linked list.
  - List manipulations are slower (because more pointers must be updated).
  - Greater chance of having bugs (because more links must be manipulated)

Applications of doubly linked list

- The cache in your browser that allows you to hit the BACK and FORWARD button (a linked list of URLs).
- UNDO/REDO functionality in Photoshop or Word (a linked list of state).
- Text editor (pageUp, pageDown), (cursor movement), etc.
- A great way to represent a deck of cards in a game.

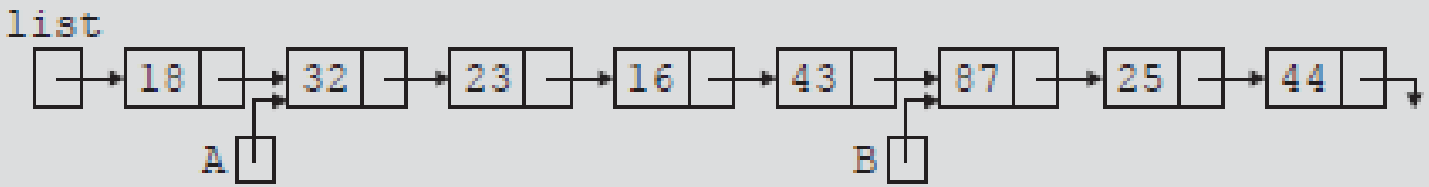
Quiz



What is the output of each of the following statement?

```
P=list;
while(p!=null){
   print(p.info+" ");
   p=p.next;
}
```

Quiz



What is the output of each of the following statement?

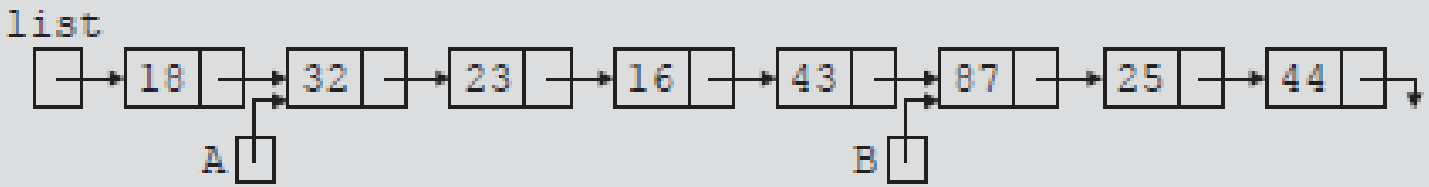
a. Print(list.data);

b. Print(A.data);

c. Print(B.next.data);

d. Print(list.next.next.data);

Quiz



What is the value of each of the following relational expression?

a. list.data>=18
b. list.next==A
c. A.next.data==16
d. B.next==null
e. list.data==18

Ans.

a. True
b. True
c. False
d. False
e. True

Quiz

- Write code to reverse the singly link list?

Suppose a linklist is 1→2→3→4

Change to 4→3→2→1 , without creating extra node.

Programming Example: Video Store

- The program should perform these operations:
  - Rent a video
  - Return a video
  - Create a list of videos owned by the store
  - Show the details of a particular video
  - Print a list of all videos in the store
  - Check whether a particular video is in the store
  - Maintain a customer database
  - Print a list of all videos rented by each customer

Examples

- Merge lists: if lists in sorted order
- Given two doubly linked lists **L1** and **L2**, write a member Boolean function to check if **L1** is the reverse of **L2**. write big-Oh

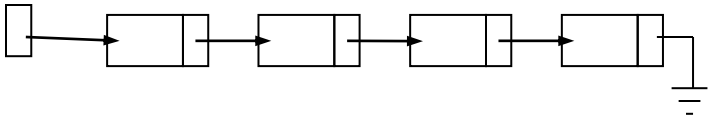
- Do the same for singly linked list and find it's big-Oh.

Analysis of doubly linked list(DLL)

- More space is required because of two references.
- Move back and forth.
- More assignment operations require to update references in insertion or deletion operation.
- insertion and deletion are independent to a size of a list.
- Big *O* in worst case:
  - Searching in DDL is *O*(n)
  - Insertion and deletion in DDL is *O*(constant)

Summary of linked lists

- Types of linked lists:
  - Singly linked list
    - Begins with a pointer to the first node
    - Terminates with a null pointer
    - Only traversed in one direction



  - Doubly linked list
    - Begins with head pointer
    - Each node has a forward pointer and a backward pointer
    - Allows traversals both forwards and backwards