TECHNO INDIA UNIVERSITY
WESTBENGAL

Data Structures & Algorithms
(PCC-CS 301)

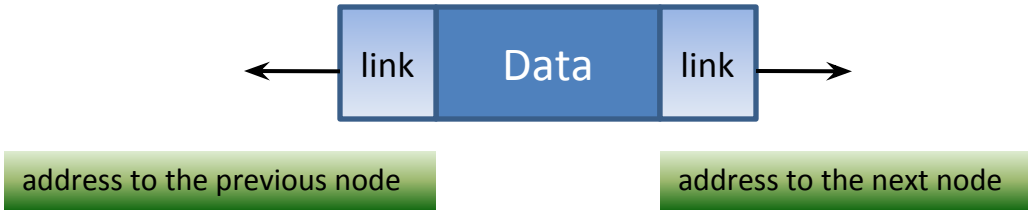
Dr. Debashis Das
Associate Professor
Department of CSE
Techno India University, Kolkata

Topics Covered

1. Doubly Linked List
    a. Representation
    b. Operations

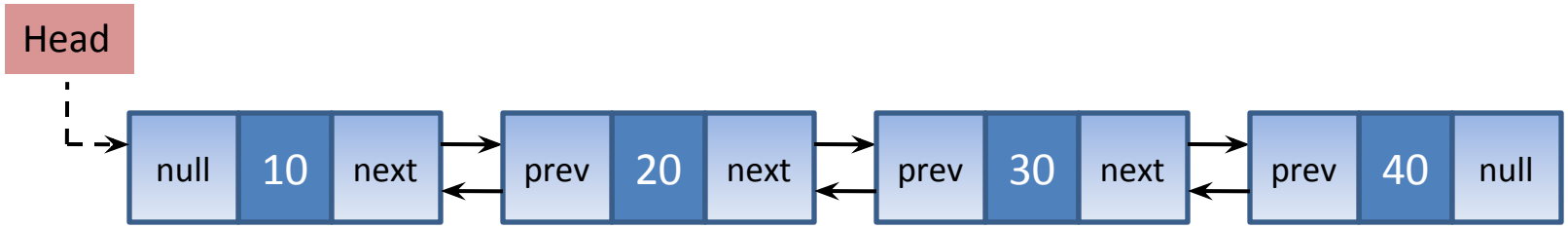
Doubly Linked List

- Representation
  - Node
    - Each node will contain 3 fields, one data field and two address fields
    - Left address field will store the address of the previous node
    - Right address field will store the address of the next node



Doubly Linked List

- Representation
  - List
    - First node will be pointing by the **Head** pointer
    - Left address field of the first node will be **null**
    - Right address field of the last node will be **null**



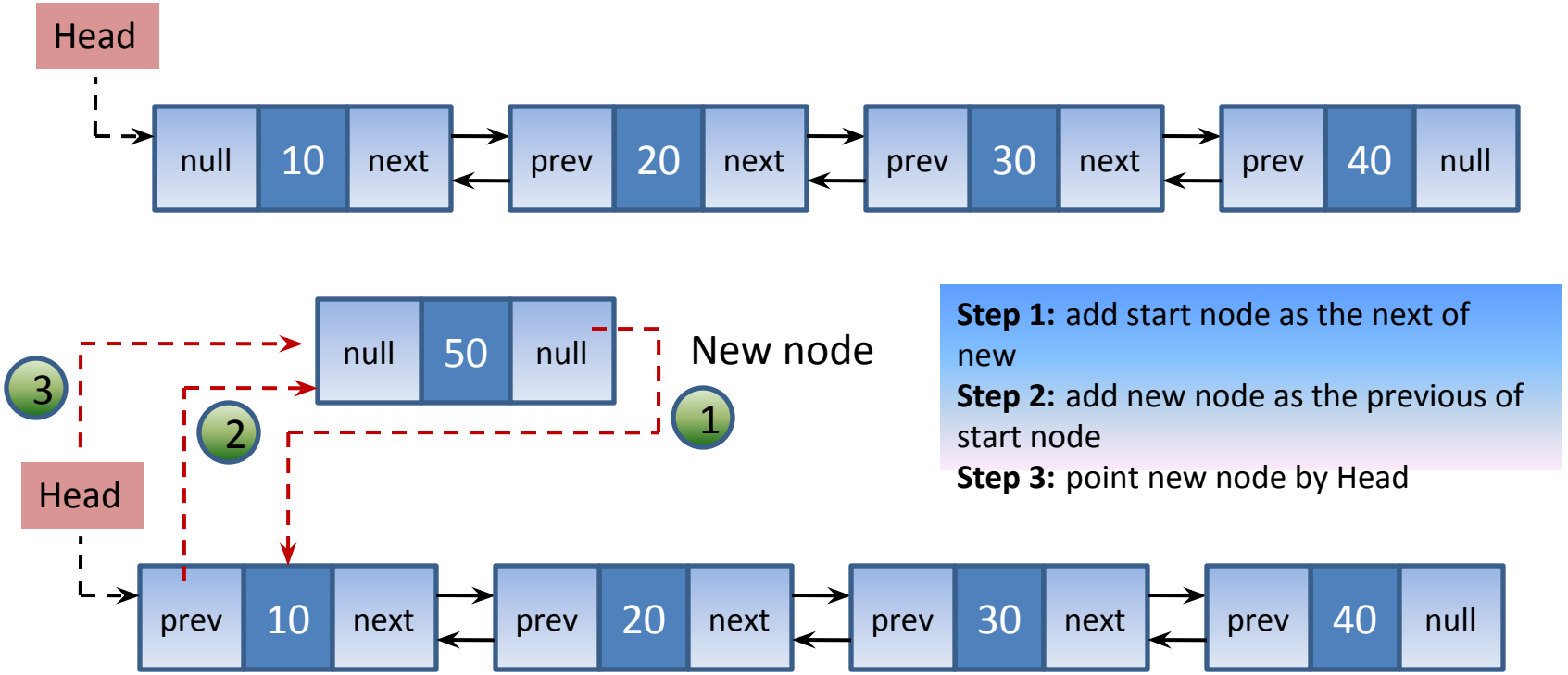
Doubly Linked List

- Operations
  - Data Insertion
    - Insertion at beginning
    - Insertion at end
    - Insertion in middle
  - Data Deletion
    - Delete start node
    - Delete last node
    - Delete any node from middle
  - Displaying the list

Data Insertion in Doubly Linked List

Doubly Linked List

- Data Insertion
  - Insert node at beginning



Step 1: add start node as the next of new
Step 2: add new node as the previous of start node
Step 3: point new node by Head

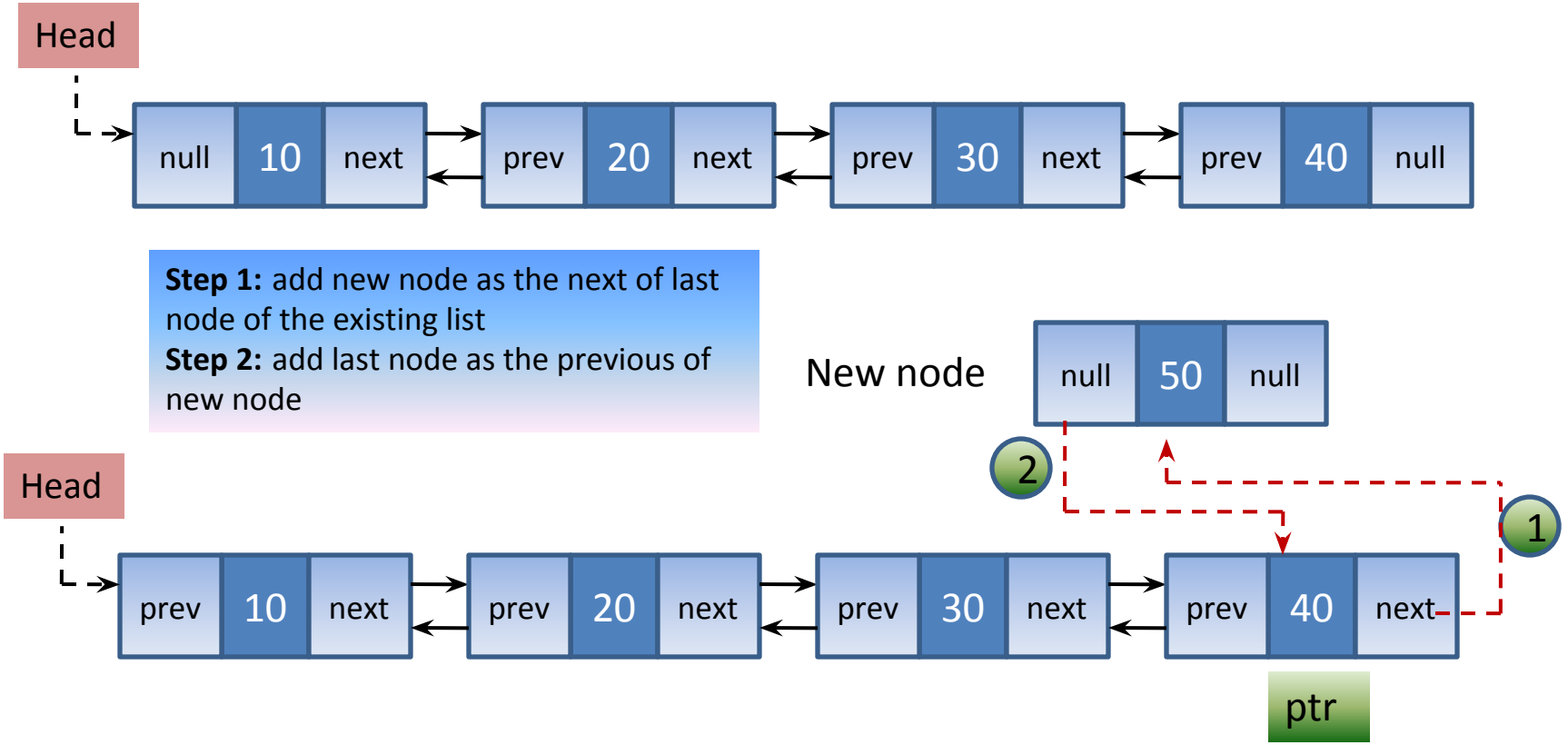
Doubly Linked List

- Data Insertion
  - Insert node at beginning (algorithm)

```
Insert_begin(LL, N)  // N is the new node to insert, LL is the existing list
{
  if LL = null
     head := N
  else
     N -> next := head
     head -> prev := N
     head := N
}
```

Doubly Linked List

- Data Insertion
  - Insert node at end



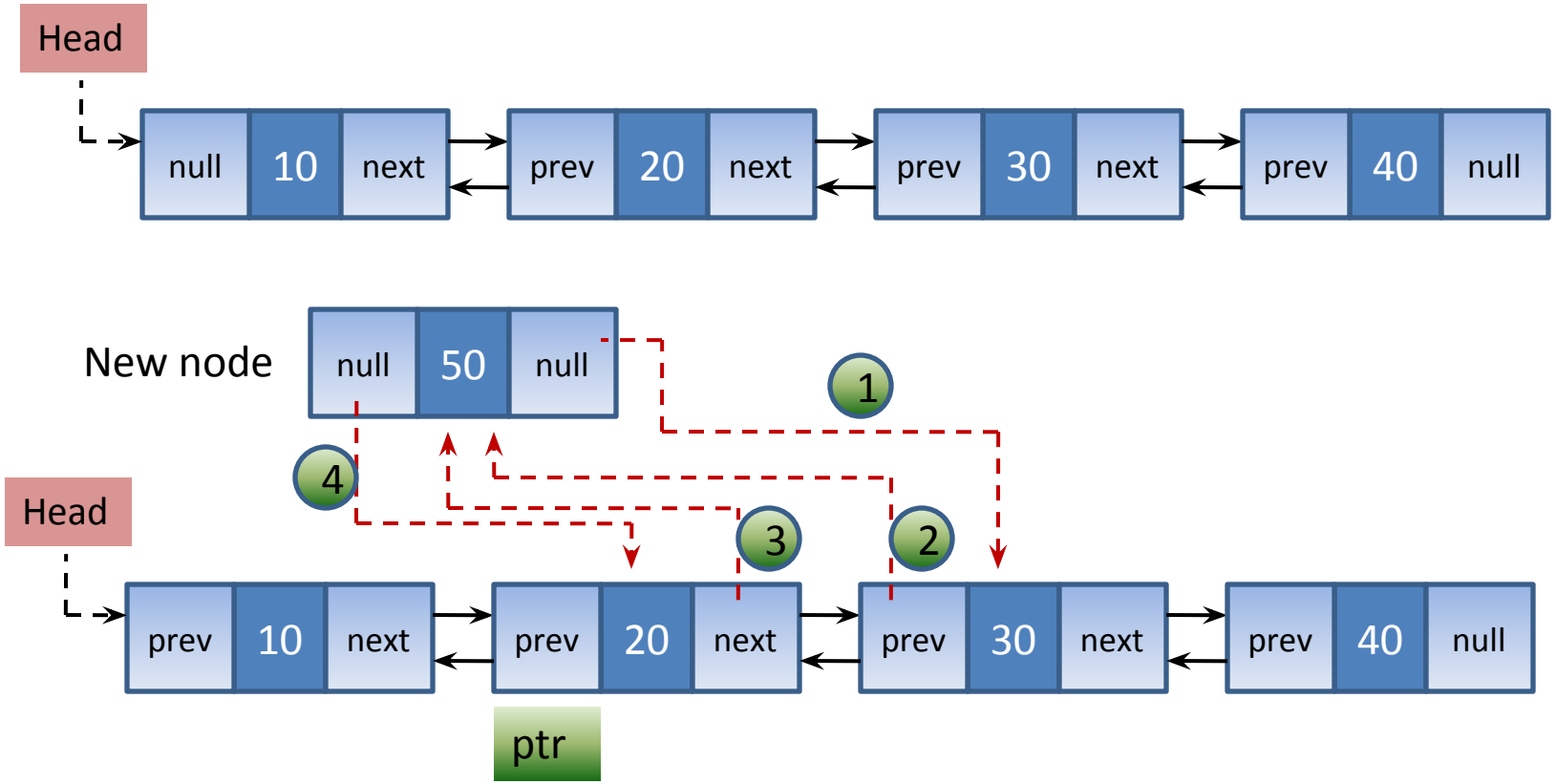
Doubly Linked List

- Data Insertion
  - Insert node at end (algorithm)

```
Insert_end(LL, N)  // N is the new node to insert, LL is the existing list
{
  if LL = null
     head := N
  else
     set ptr := head
     while ptr -> next != null
        ptr := ptr -> next
     ptr -> next := N
     N -> prev := ptr
}
```

Doubly Linked List

- Data Insertion
  - Insert in middle (insert after data 20)



Doubly Linked List

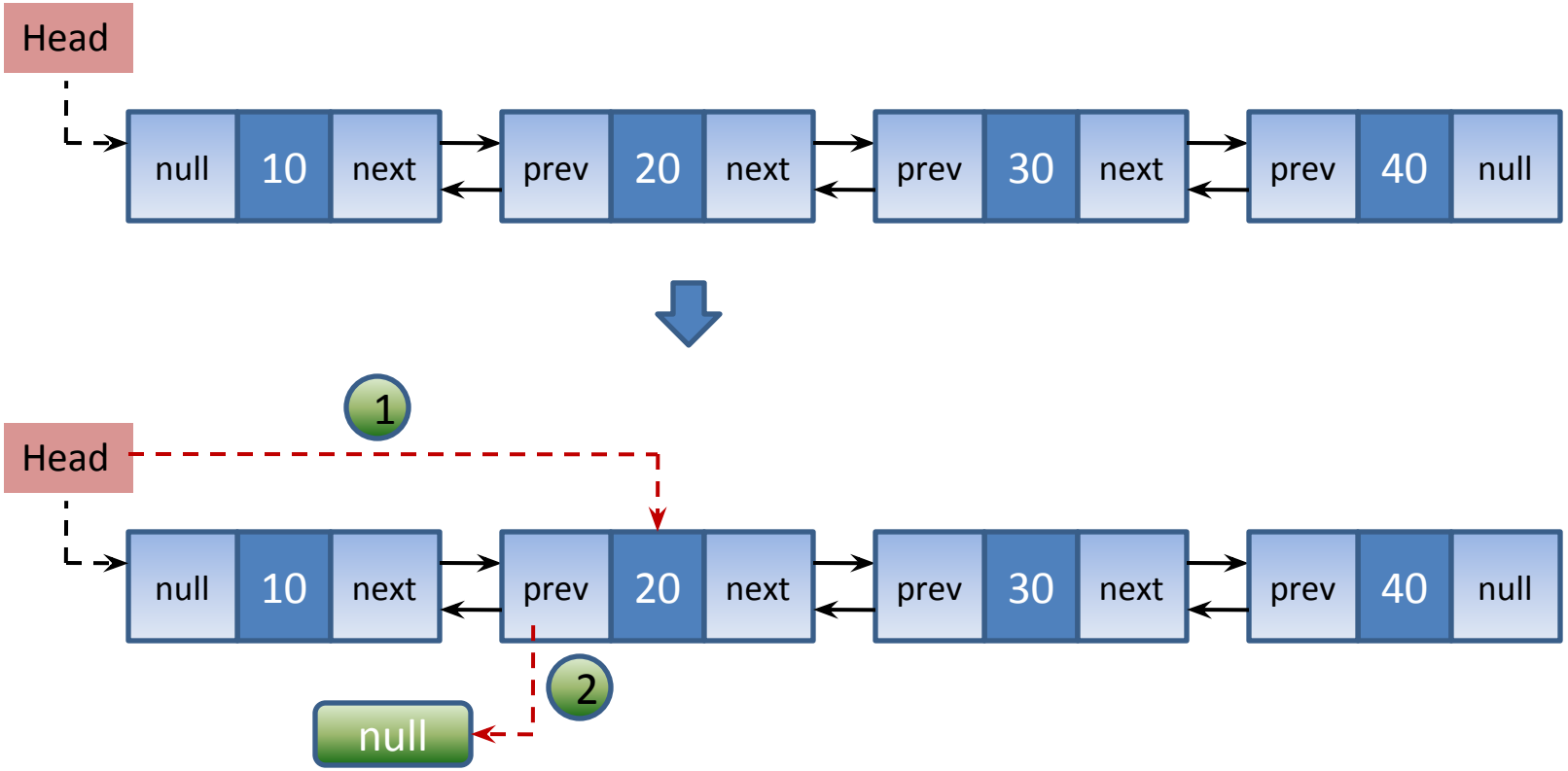
- Data Insertion
  - Insert node in middle (algorithm)

```
Insert_middle(LL, N, x)  // N is the new node to insert, LL is the existing list
{                        // x is the data after which new node will be inserted
 if LL = null
   head := N
 else
   set ptr := head
   while ptr -> data != x
      ptr := ptr -> next
   N -> next := ptr -> next
   ptr -> next -> prev := N
   ptr -> next := N
   N ->prev := ptr
}
```

Data Deletion from Doubly Linked List

Doubly Linked List

- Data Deletion
  - Delete first node



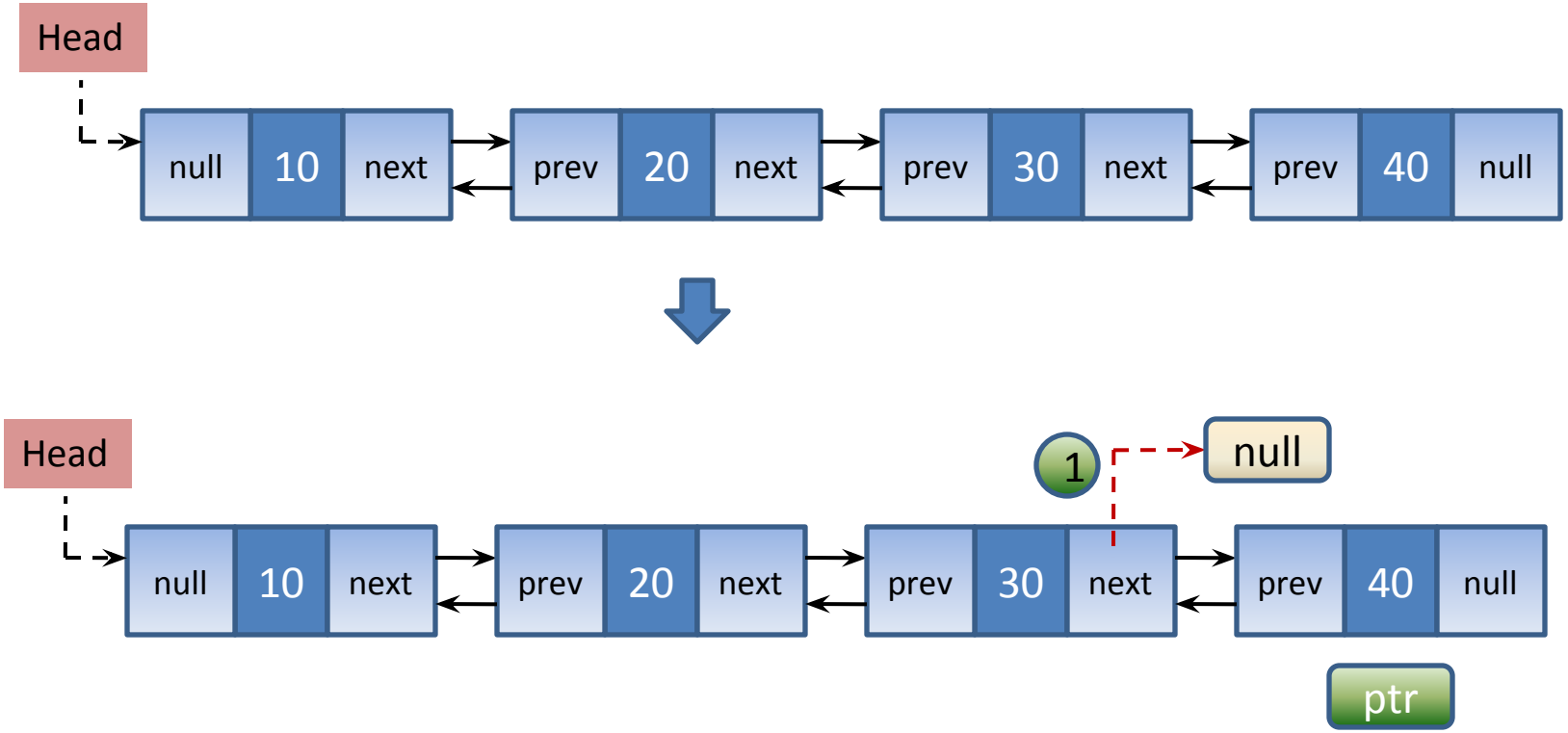
Doubly Linked List

- Data Deletion
  - Delete first node (algorithm)

```
Delete_begin(LL)  // LL is the existing list
{
  if head->next = null
     head := null
  else
      head := head->next
      head->prev := null
}
```

Doubly Linked List

- Data Deletion
  - Delete last node



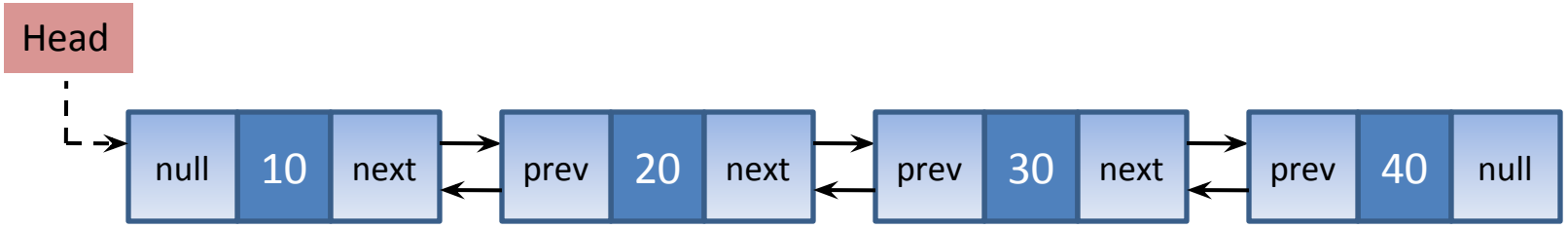
Doubly Linked List

- Data Deletion
  - Delete last node(algorithm)

```
Delete_end(LL)  // LL is the existing list
{
  if head->next = null
     head := null
  else
     set ptr := head
      while ptr->next != null
           ptr := ptr->next
     ptr->prev->next := null
     free (ptr)
}
```

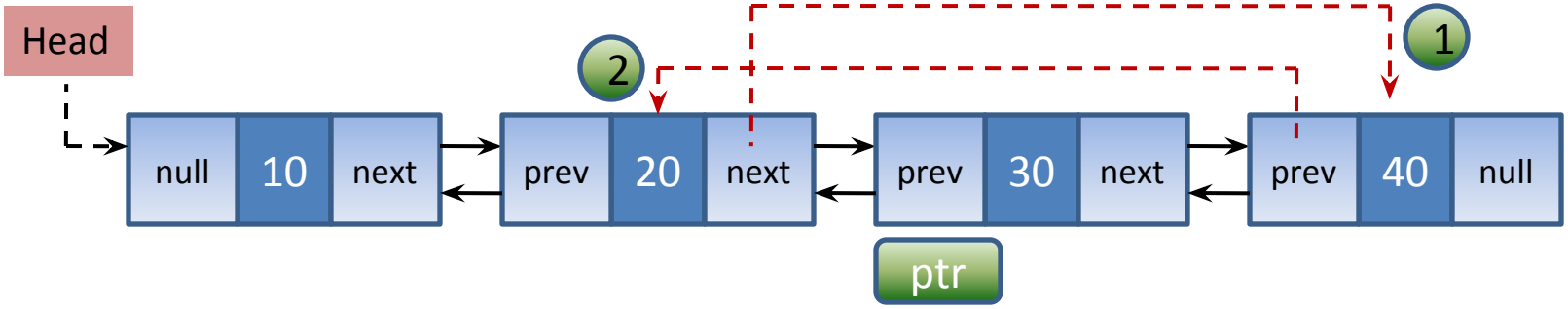
Doubly Linked List

- Data Deletion
  - Delete node from middle



Delete 30

Operation 1 & 2 can be performed in any sequence



Doubly Linked List

- Data Deletion
  - Delete node from middle (algorithm)

```
Delete_middle(LL,x)  // LL is the existing list and 'x' is the data to delete
{
  if head->next = null
     head := null
  else
     set ptr := head
      while ptr->data != x
           ptr := ptr->next
     ptr->prev->next := ptr->next
     ptr->next->prev := ptr->prev
     free (ptr)
}
```

Data Display of Doubly Linked List

Doubly Linked List

- Display Linked List
  - Display the list (algorithm)

```
Display(LL)  // LL is the existing list
{
  if LL = null
     Print "Empty list" and return
  else
     set ptr := head          // ptr is a pointer set to head of the list
     while ptr not equal to null   // traversing the entire list
              Print ptr->data
              ptr := ptr->next
}
```

Queries?