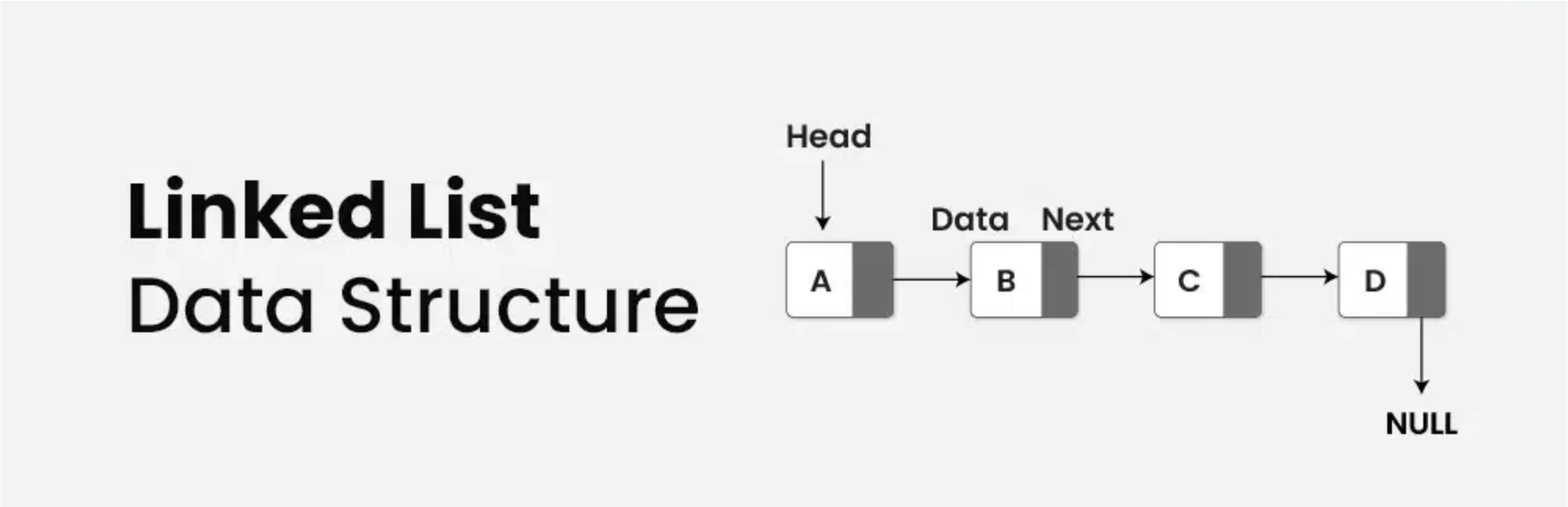
Programming Practice for Data Science

Lecture 2: Linked List (9/20/24)

Taesup Kim
Graduate School of Data Science
Seoul National University

Linked List

Definition



Linked List

Definition

```python
class Node:
    def __init__(self, data):
        self.data = data  # 노드가 저장하는 데이터
        self.next = None  # 다음 노드를 가리키는 포인터
```

```cpp
#include <iostream>
using namespace std;

class Node {
public:
    int data; // 노드가 저장하는 데이터
    Node* next; // 다음 노드를 가리키는 포인터

    Node(int data) {
        this->data = data;
        this->next = nullptr;
    }
};
```

Linked List

Definition

- **Dynamic Data structure**
  - the size of memory can be allocated or de-allocated at run time (based on the operation insertion or deletion)
- **Ease of Insertion/Deletion**
  - insertion and deletion of elements are simpler than arrays since no elements need to be shifted after insertion and deletion (just the address needed to be updated)

Linked List

Definition

- **Efficient Memory Utilization**
  - the size increases or decreases as per the requirement so this avoids the wastage of memory
- **Implementation**
  - Various advanced data structures can be implemented using a linked list like a stack, queue, graph, hash maps, etc.

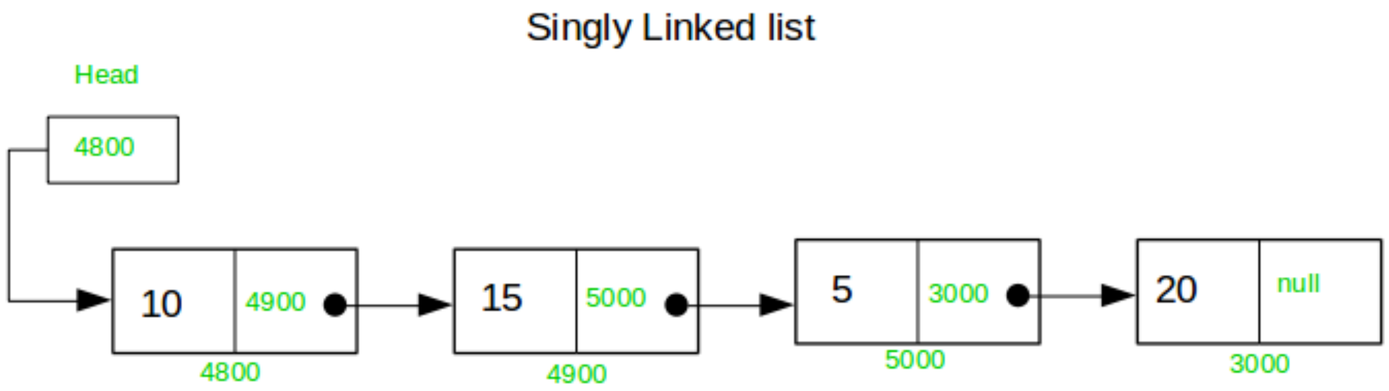
Singly Linked List

Types

- Singly Linked List
- Doubly Linked List
- Circular Linked List

Singly Linked List

Definition



LAAL

Singly Linked List

Definition

Singly Linked List

Basic Operations

- Initialization
- Insertion
- Search
- Delete
- Reverse
- ...

Singly Linked List

Basic Operations

- Initialization
- Insertion
- Search
- Delete
- Reverse
- ...

Singly Linked List

Basic Operations

- **Search**

Input: *14 -> 21 -> 11 -> 30 -> 10, key = 14*
Output: *Yes*
Explanation: *14 is present in the linked list.*

Input: *6 -> 21 -> 17 -> 30 -> 10 -> 8, key = 13*
Output: *No*
Explanation: *No node in the linked list has value = 13.*

Singly Linked List

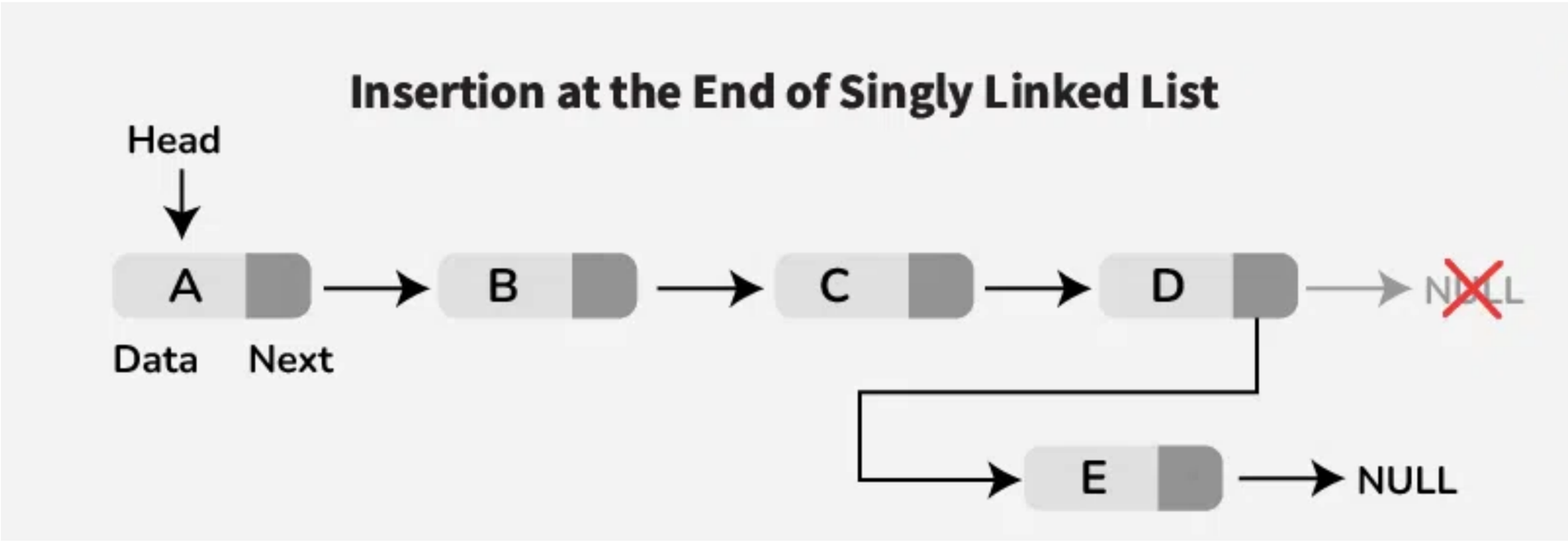
Basic Operations

- Initialization
- Insertion
- Search
- Delete
- Reverse
- ...

Singly Linked List

Basic Operations

- **Insertion (at the end)**



Singly Linked List

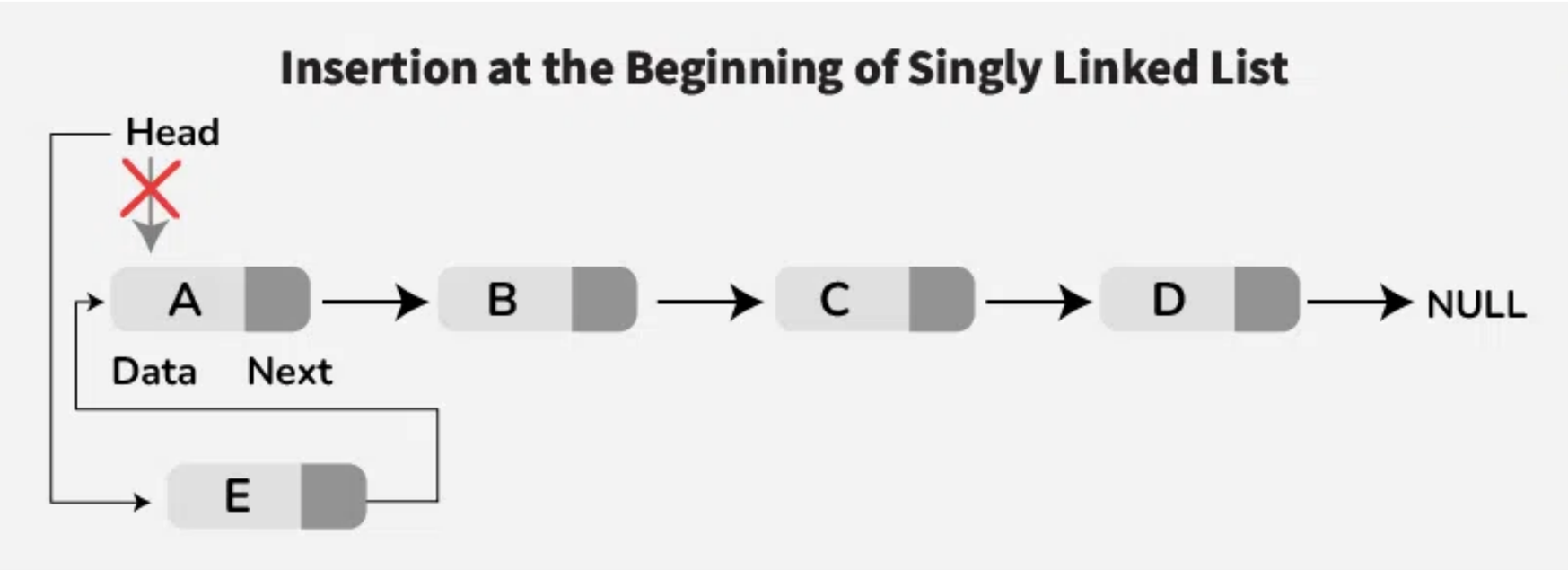
Basic Operations

- **Insertion (at the end)**
  - new node
  - Go to the last node of the Linked List
  - Change the next pointer of last node from NULL to the new node
  - Make the next pointer of new node as NULL to show the end of Linked List

Singly Linked List

Basic Operations

- **Insertion (at the front)**



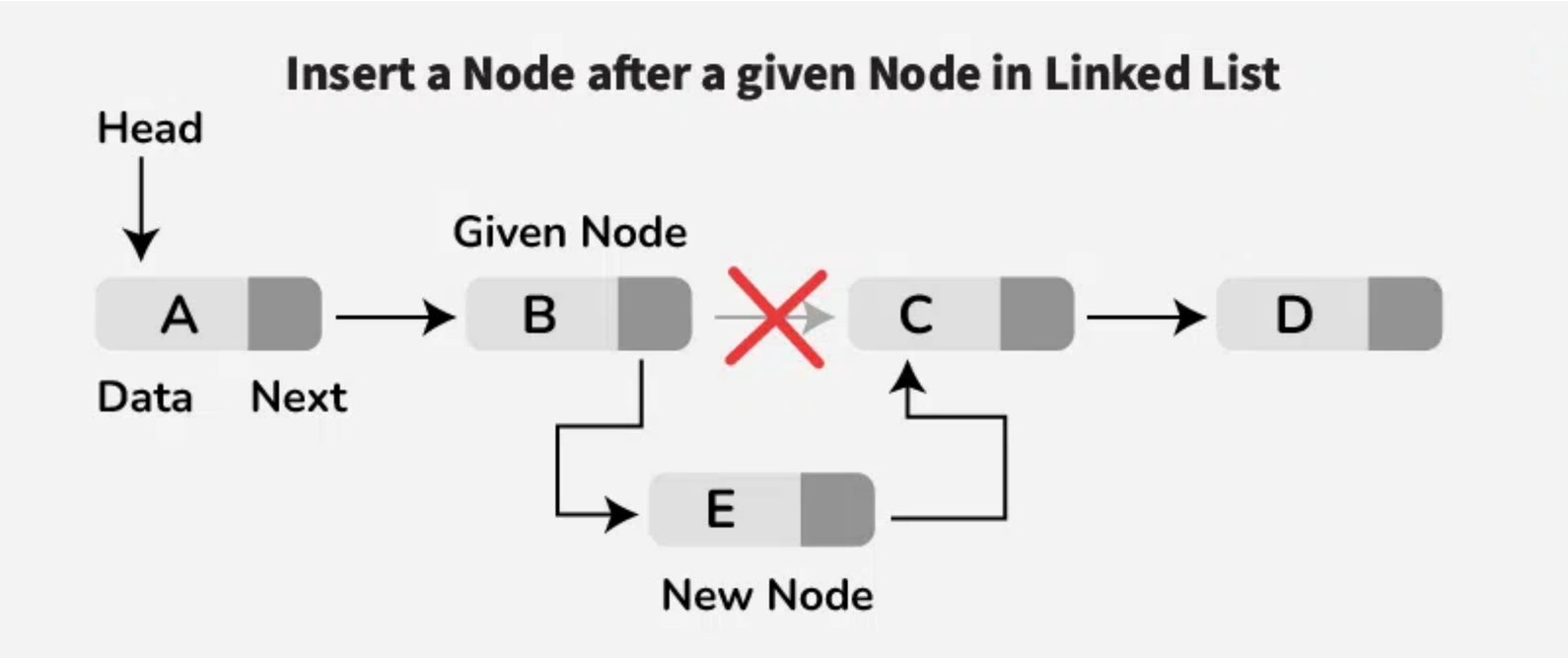
Singly Linked List
Basic Operations

- **Insertion (at the front)**
  - Make the first node of Linked List linked to the new node
  - Remove the head from the original first node of Linked List
  - Make the new node as the Head of the Linked List

Singly Linked List

Basic Operations

- **Insertion (after a given node)**



Singly Linked List

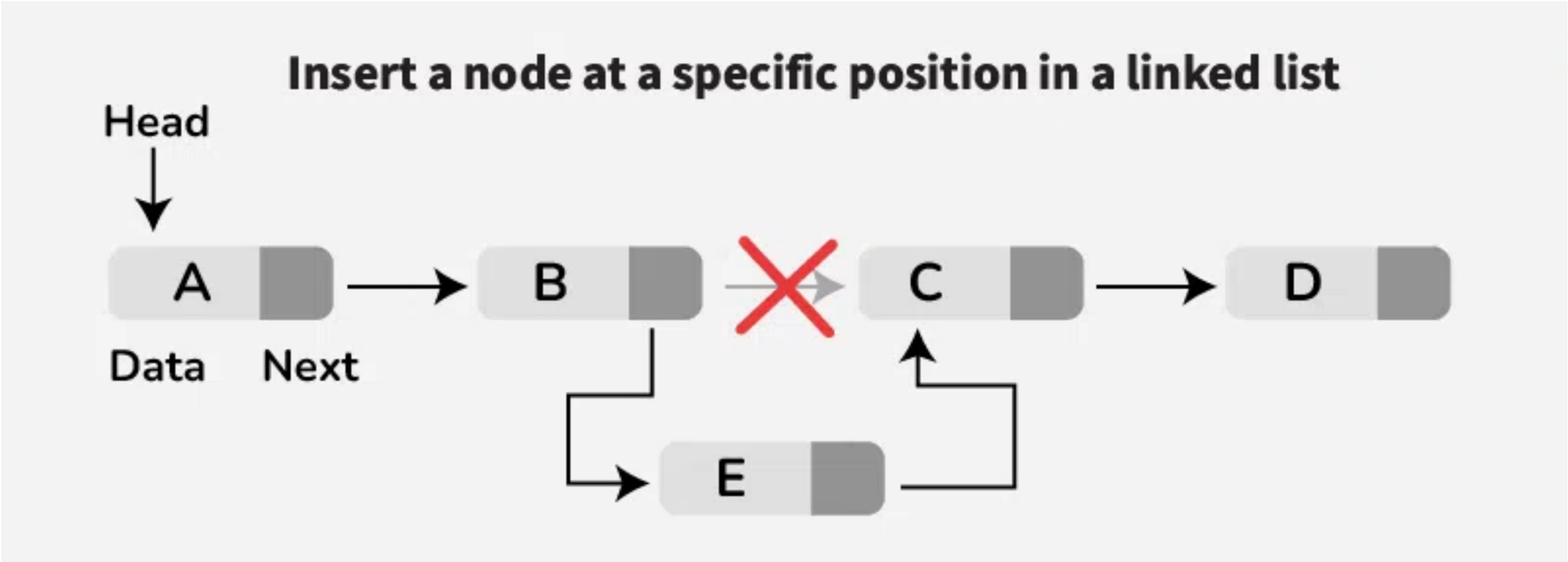
Basic Operations

- **Insertion (after a given node)**
  - Initialize a pointer curr to traverse the list starting from head
  - Loop through the list to find the node with data equal to key
    - If not found then return from function
  - Create a new node, say new_node initialized with the given data
  - Make the next pointer of new_node as next of given node
  - Update the next pointer of given node point to the new_node

Singly Linked List

Basic Operations

- **Insertion (at a specific position)**



Singly Linked List

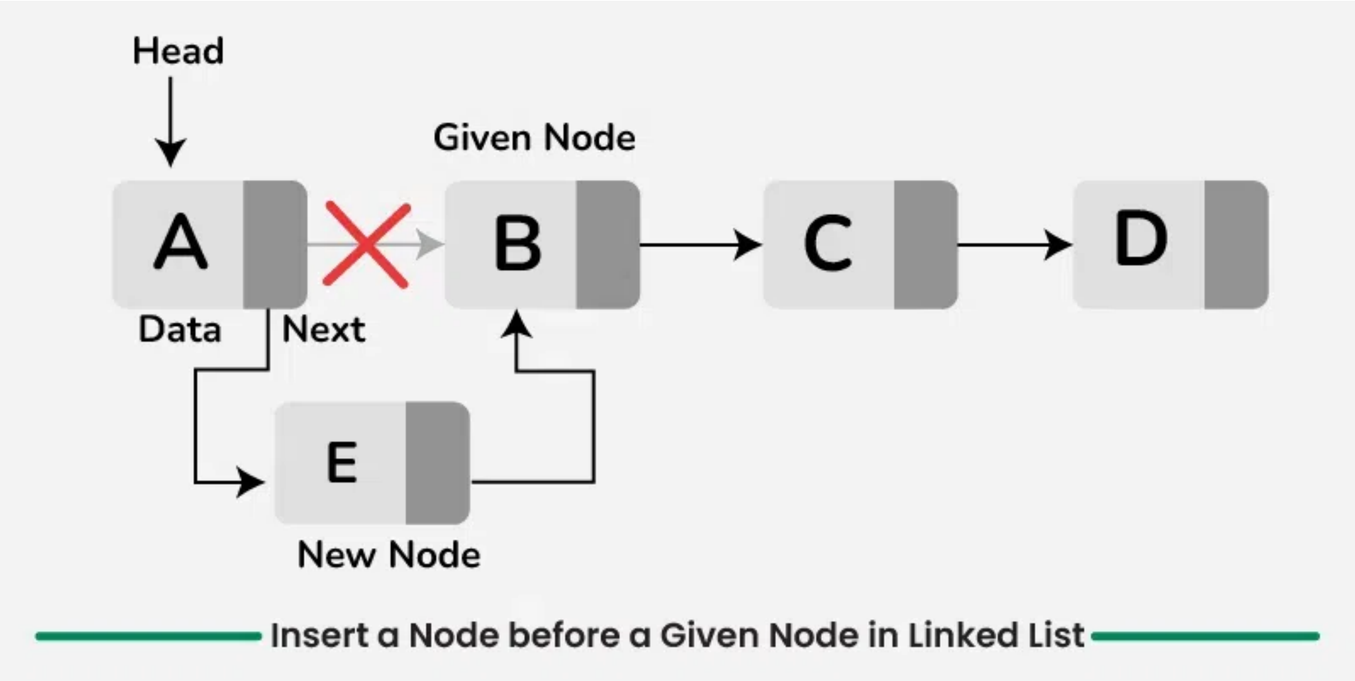
Basic Operations

- **Insertion (at a specific position)**
  - Traverse the Linked list upto position-1 nodes
  - Once all the position-1 nodes are traversed, allocate memory and the given data to the new node
  - Point the next pointer of the new node to the next of current node
  - Point the next pointer of current node to the new node

Singly Linked List

Basic Operations

- **Insertion (before a given node)**



Singly Linked List

Basic Operations

- **Insertion (before a given node)**
  - Traverse the linked list while keeping track of the previous node until given node is reached
  - Once node is found, allocate memory for a new node and set according to given data
  - Point the next pointer of the new node to node given node
  - Point the next pointer of the previous node to the new node
  - If given key is the head, update the head to point to the new node

Singly Linked List

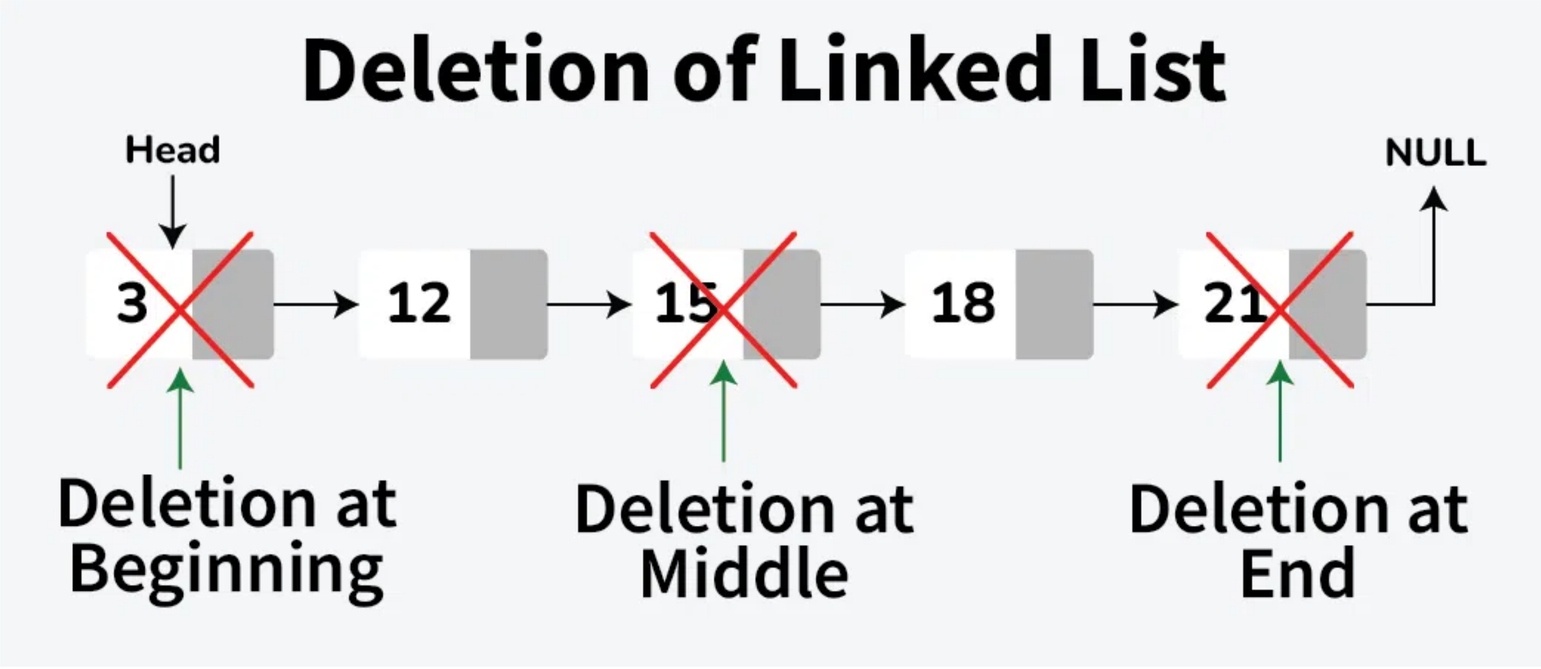
Basic Operations

- Initialization
- Insertion
- Search
- Delete
- Reverse
- ...

Singly Linked List

Basic Operations

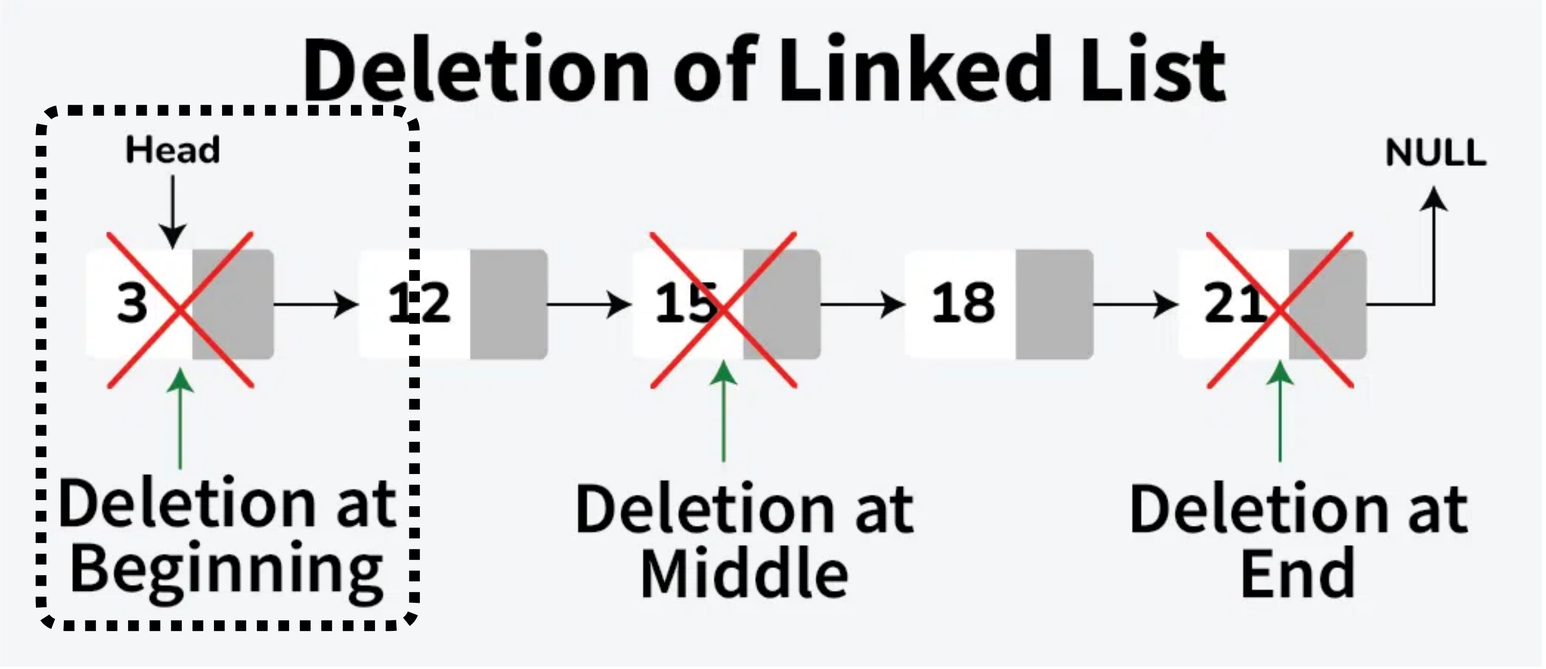
- **Delete**



Singly Linked List

Basic Operations

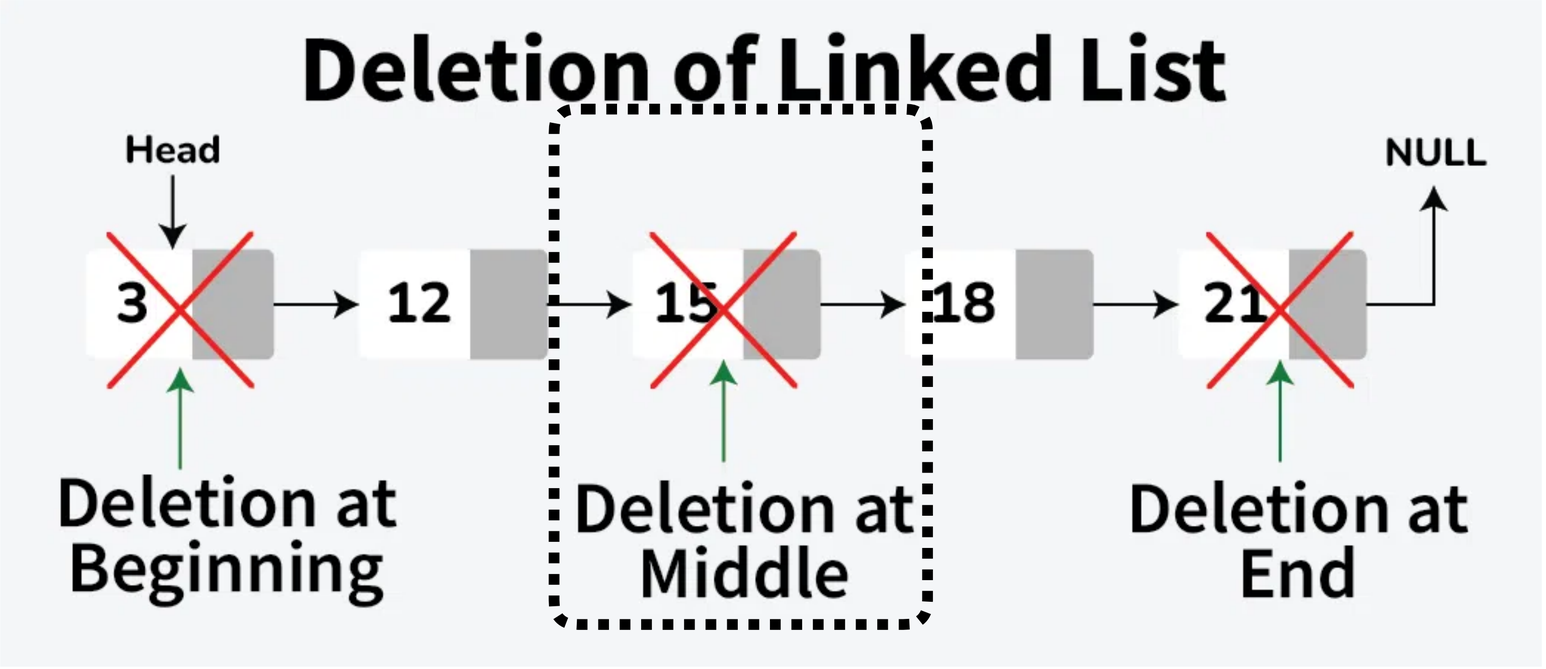
- **Delete (at beginning)**



Singly Linked List

Basic Operations

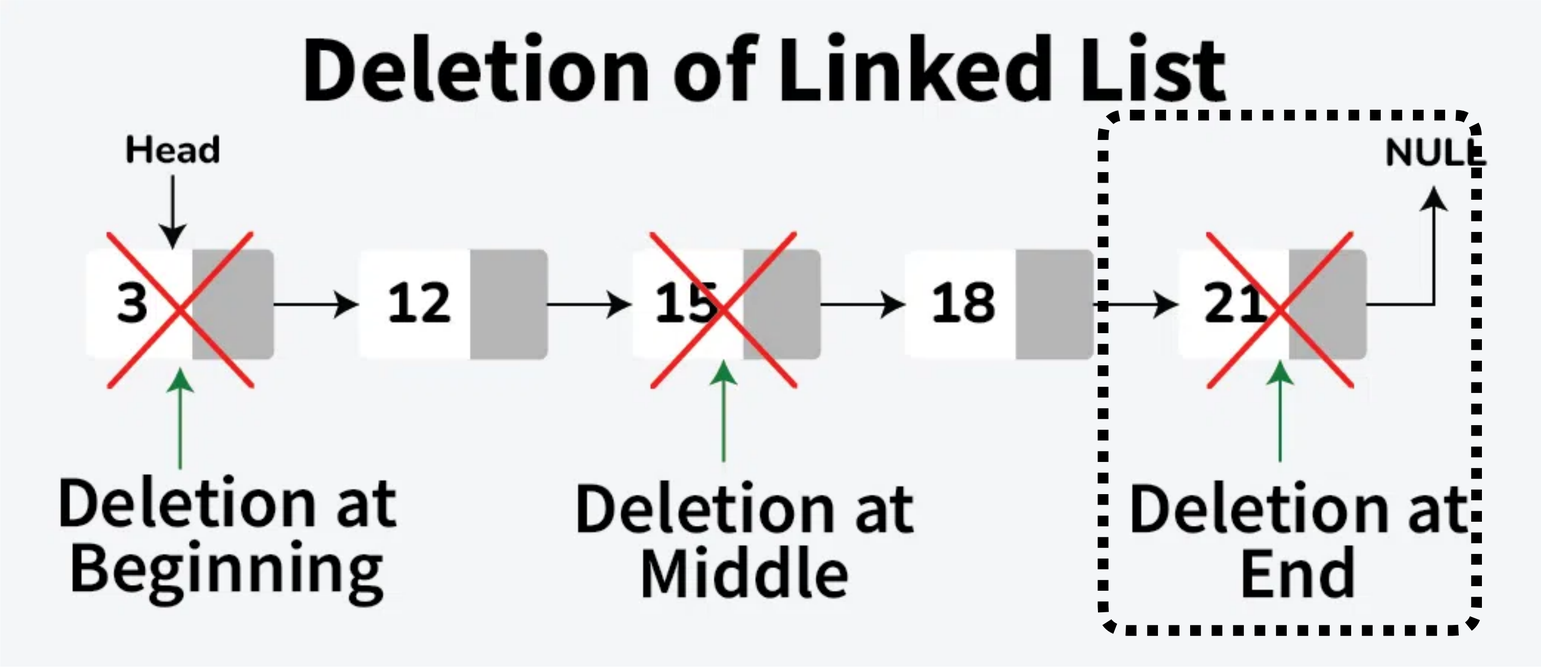
- **Delete (at middle)**



Singly Linked List

Basic Operations

- **Delete (at end)**



Singly Linked List

Basic Operations

- Initialization
- Insertion
- Search
- Delete
- Reverse
- ...

Singly Linked List

Basic Operations

- **Reverse**

Input: *Linked List = 1 -> 2 -> 3 -> 4 -> NULL*
Output: *Reversed Linked List = 4 -> 3 -> 2 -> 1 -> NULL*

Input: *Linked List = 1 -> 2 -> 3 -> 4 -> 5 -> NULL*
Output: *Reversed Linked List = 5 -> 4 -> 3 -> 2 -> 1 -> NULL*

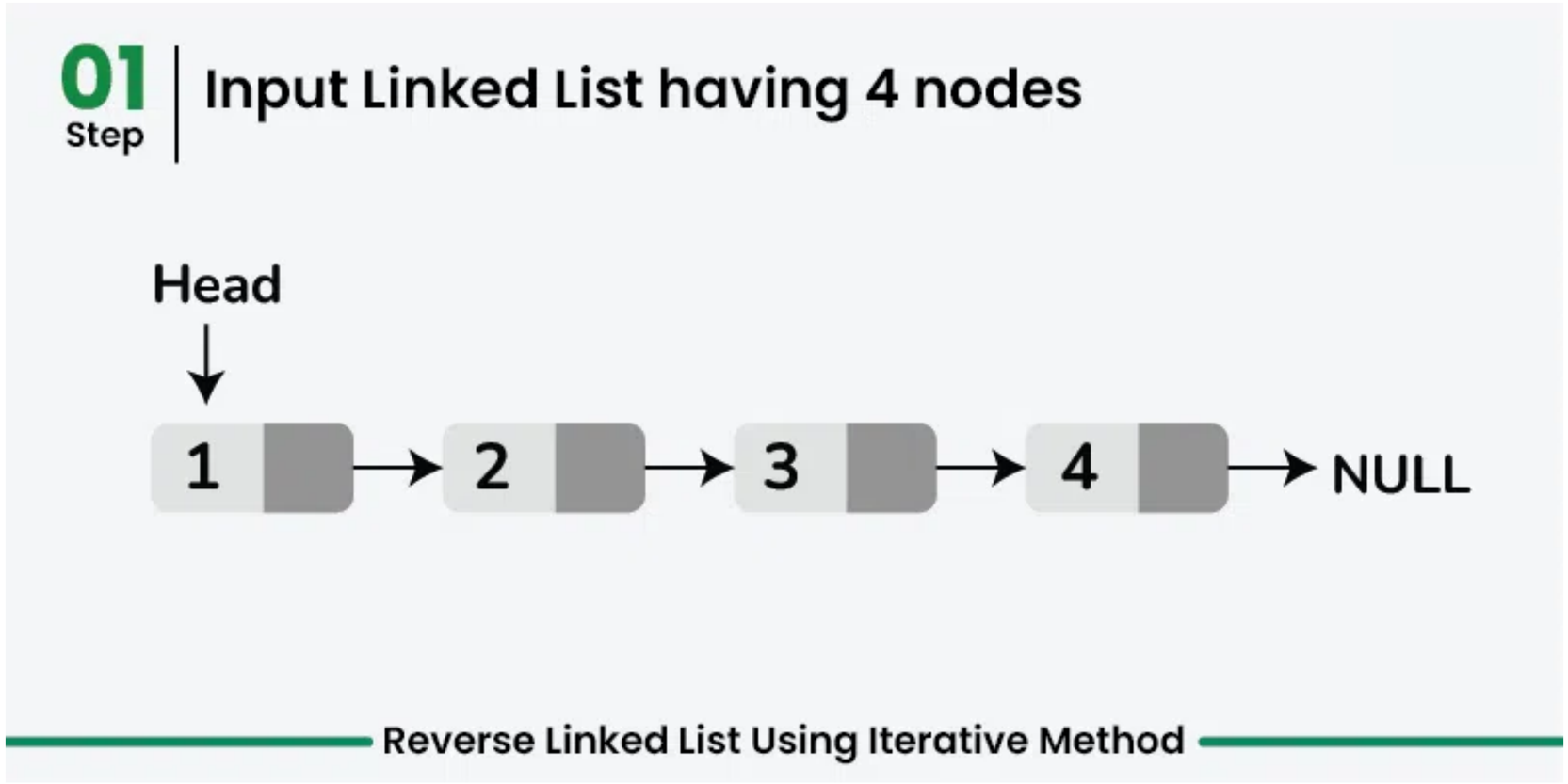
Input: *Linked List = NULL*
Output: *Reversed Linked List = NULL*

Input: *Linked List = 1->NULL*
Output: *Reversed Linked List = 1->NULL*

Singly Linked List

Basic Operations

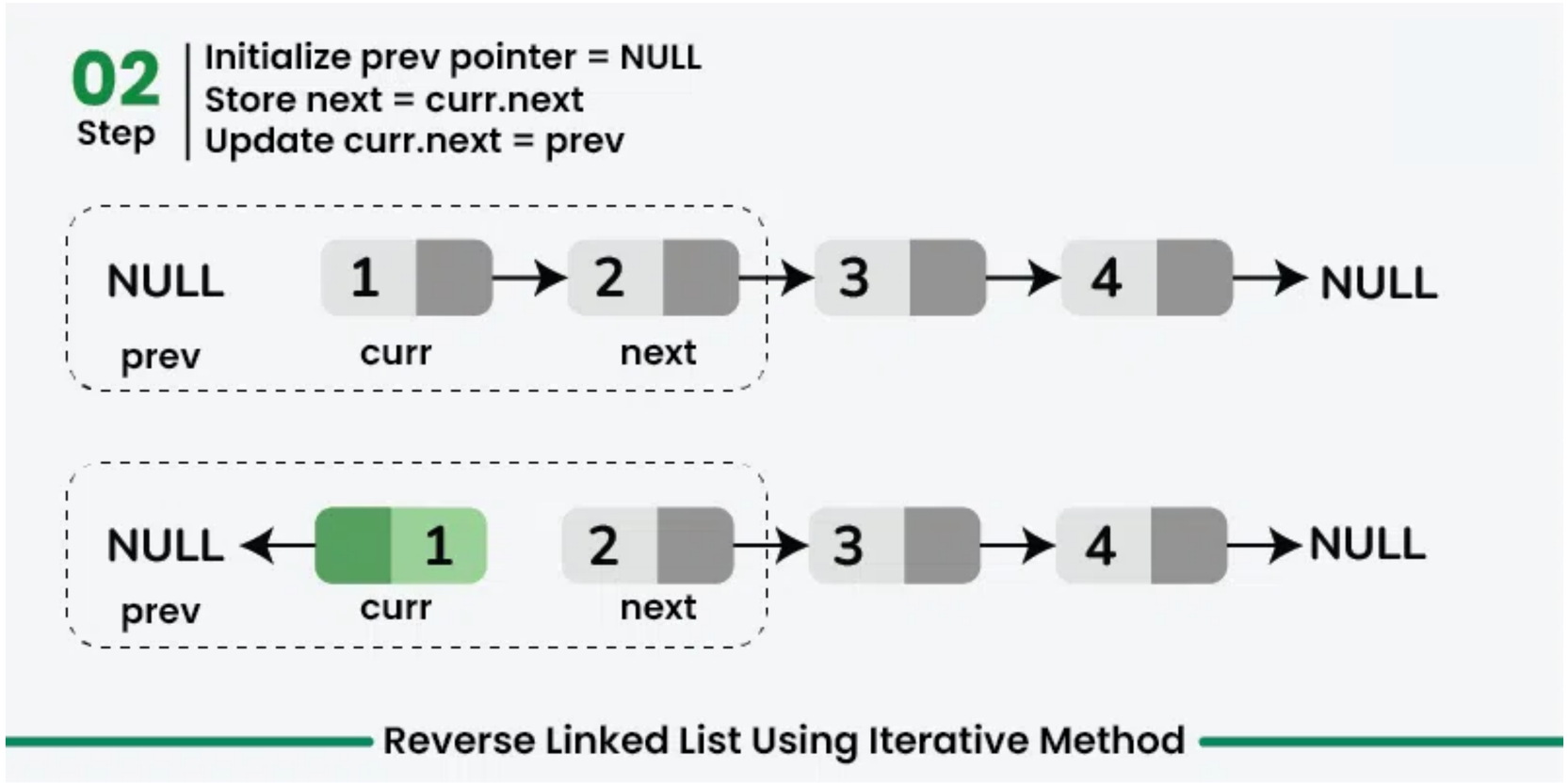
- **Reverse**



Singly Linked List

Basic Operations

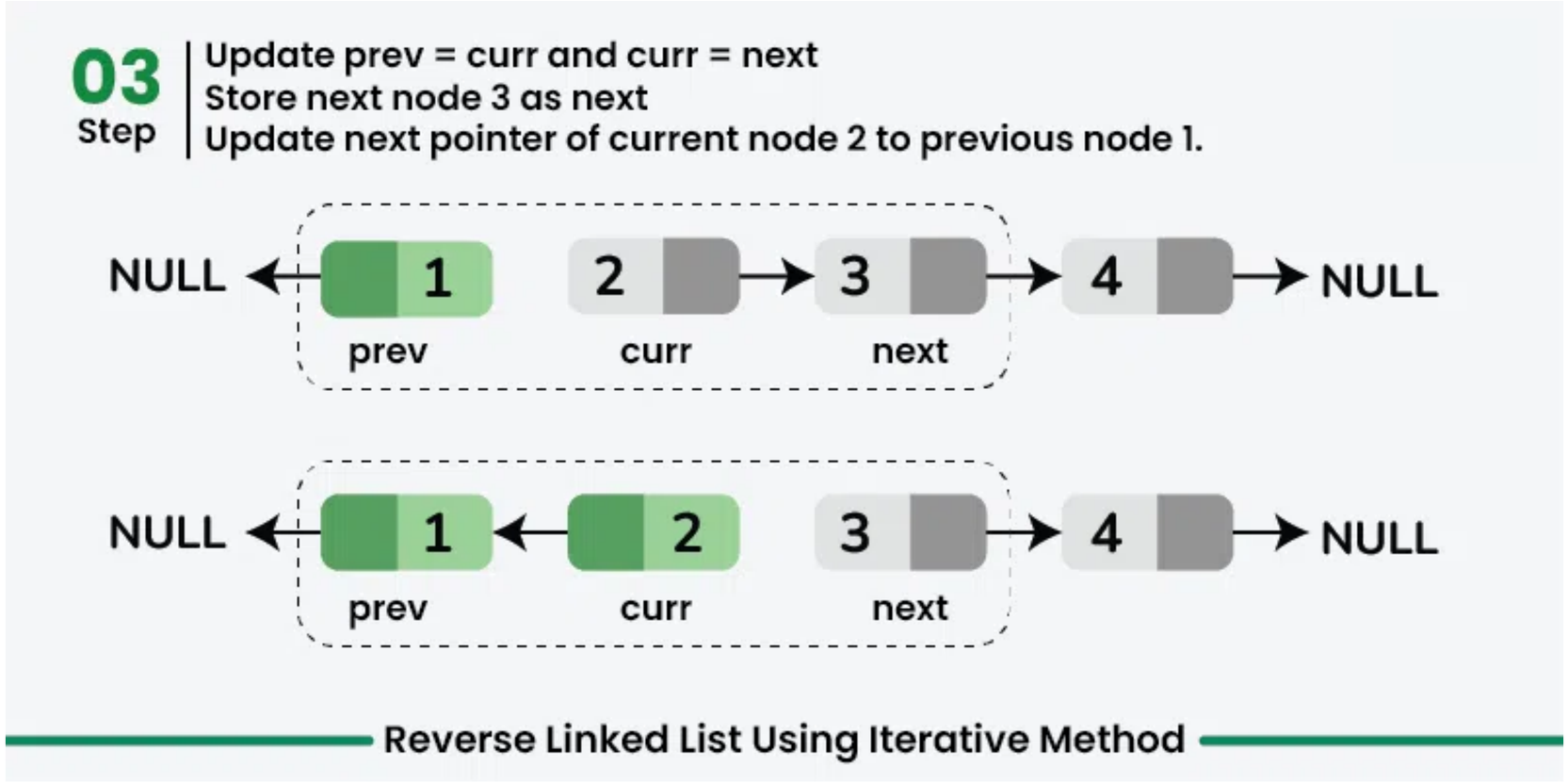
- **Reverse**



Singly Linked List

Basic Operations

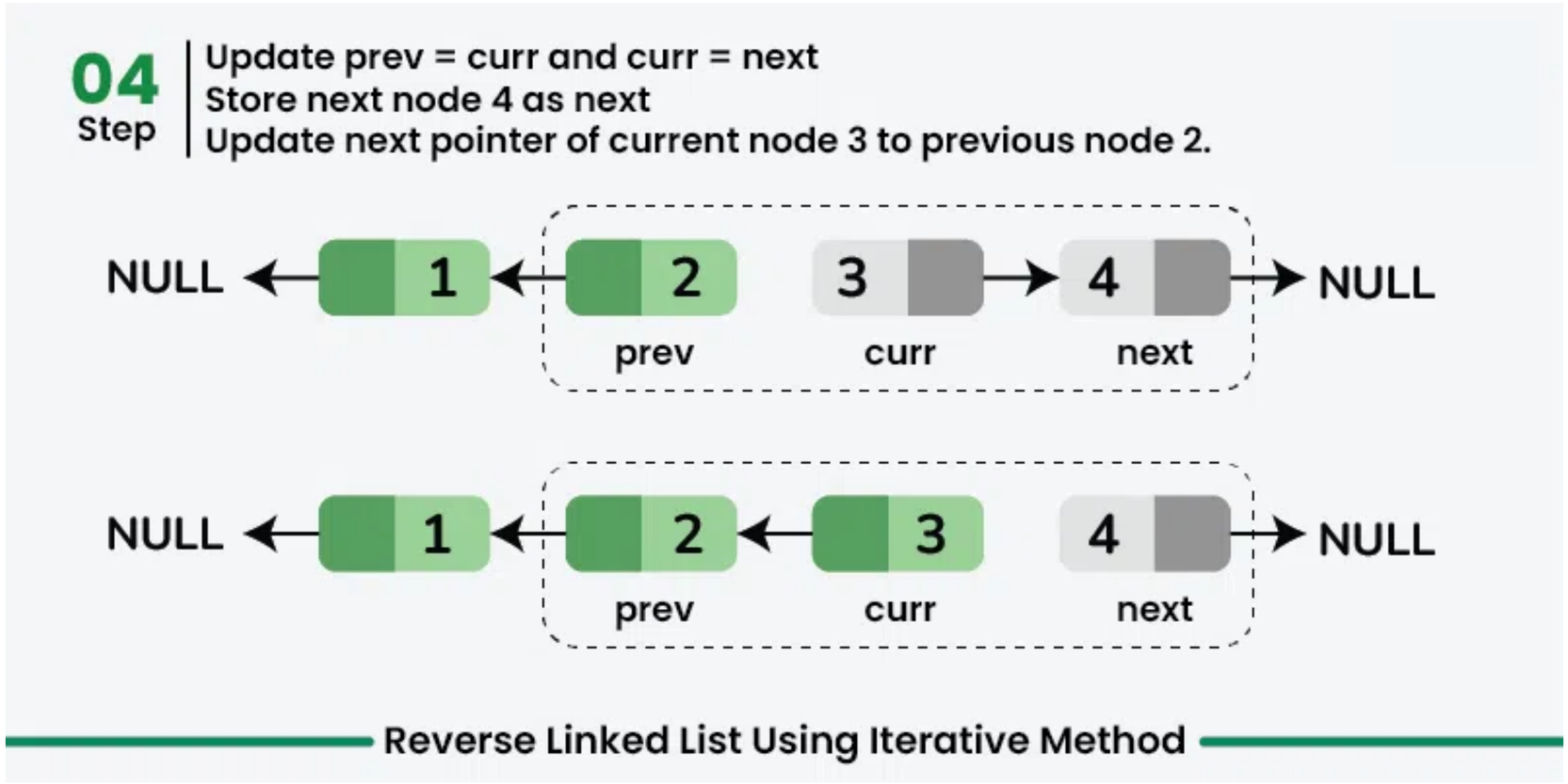
- **Reverse**



Singly Linked List

Basic Operations

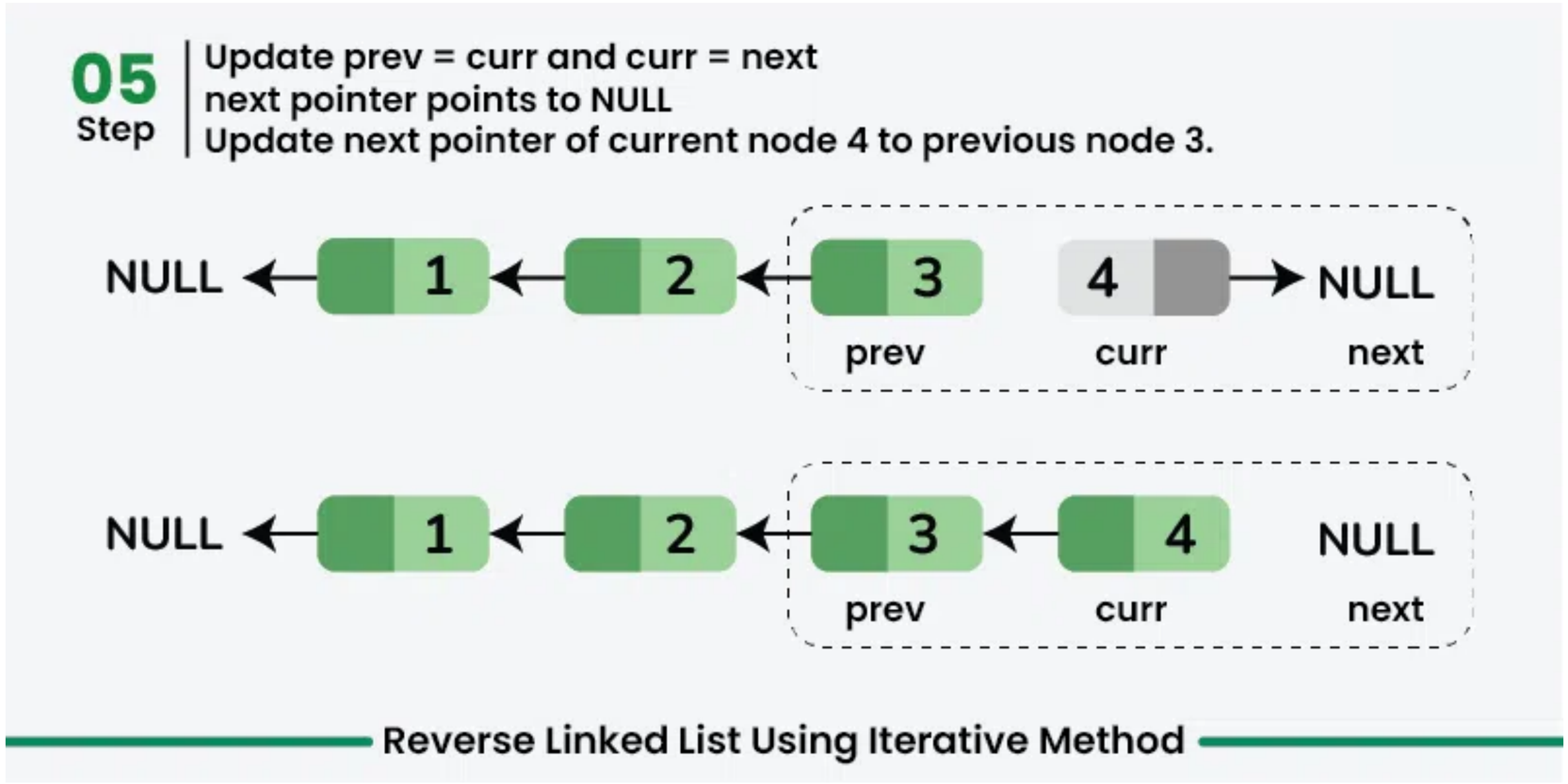
- **Reverse**



Singly Linked List

Basic Operations

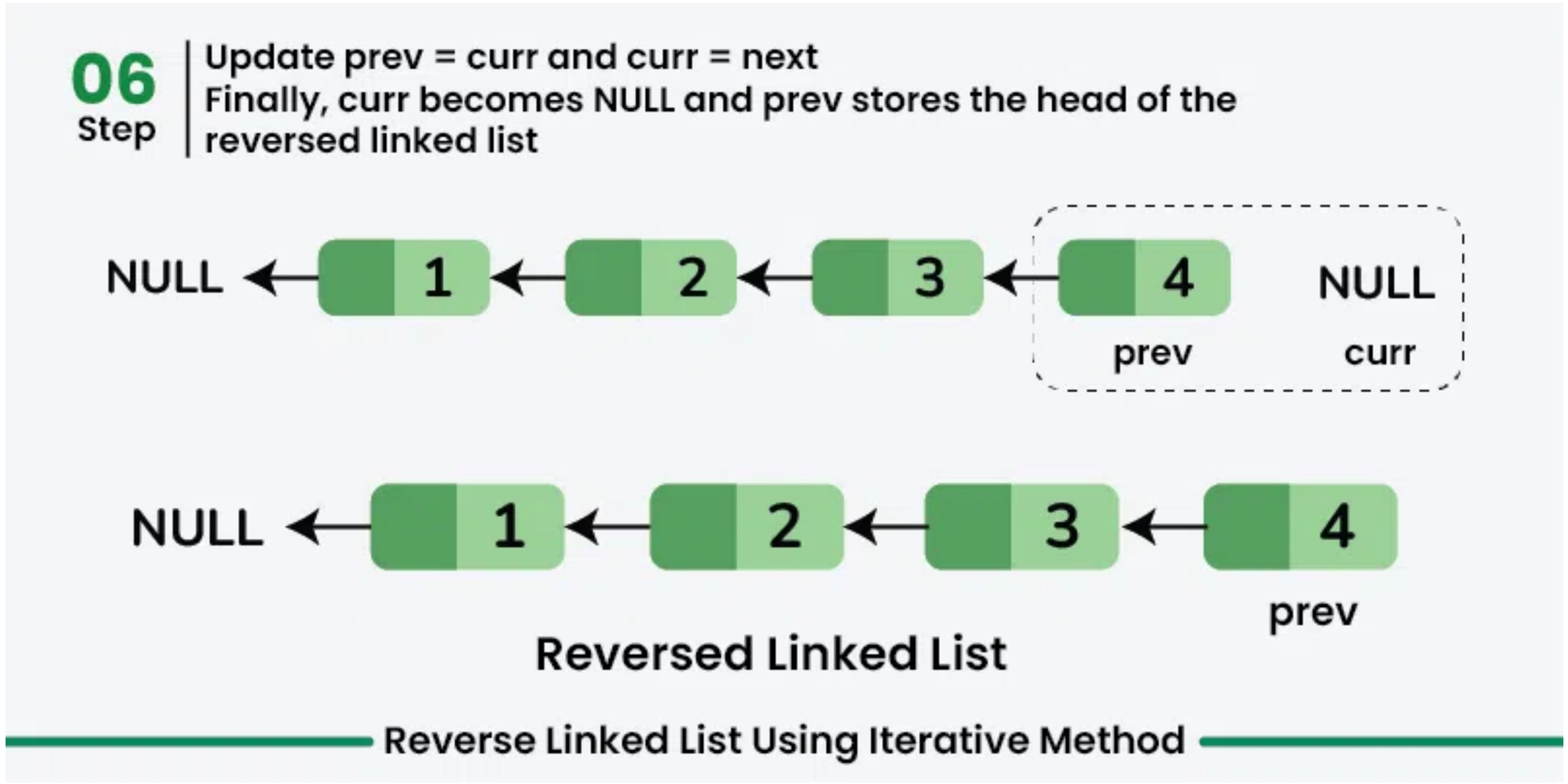
- **Reverse**



Singly Linked List

Basic Operations

- **Reverse**



Singly Linked List

Basic Operations

- **Write a function to delete a Linked List**

Singly Linked List

Basic Operations

- **Write a function to delete a Linked List**

```python
# Python program to delete a linked list

class Node:
    def __init__(self, x):
        self.data = x
        self.next = None


if __name__ == "__main__":

    # Create a hard-coded linked list:
    # 1 -> 2 -> 3 -> 4 -> 5
    head = Node(1)
    head.next = Node(2)
    head.next.next = Node(3)
    head.next.next.next = Node(4)
    head.next.next.next.next = Node(5)

    # Set head to None to remove the reference to the linked list.
    # This allows Python's garbage collector to automatically reclaim
    # the memory used by the nodes, as there are no more references
    # to the nodes in the linked list.
    head = None
    print("NULL")
```

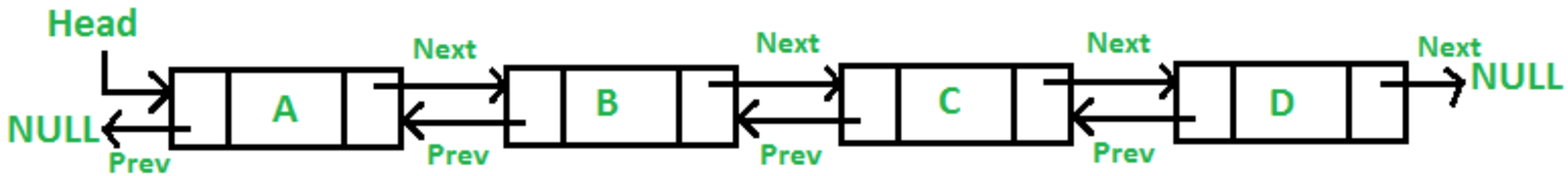
Singly Linked List

Basic Operations

- **Write a function to delete a Linked List**
  - In C++

Doubly Linked List

Definition



Doubly Linked List

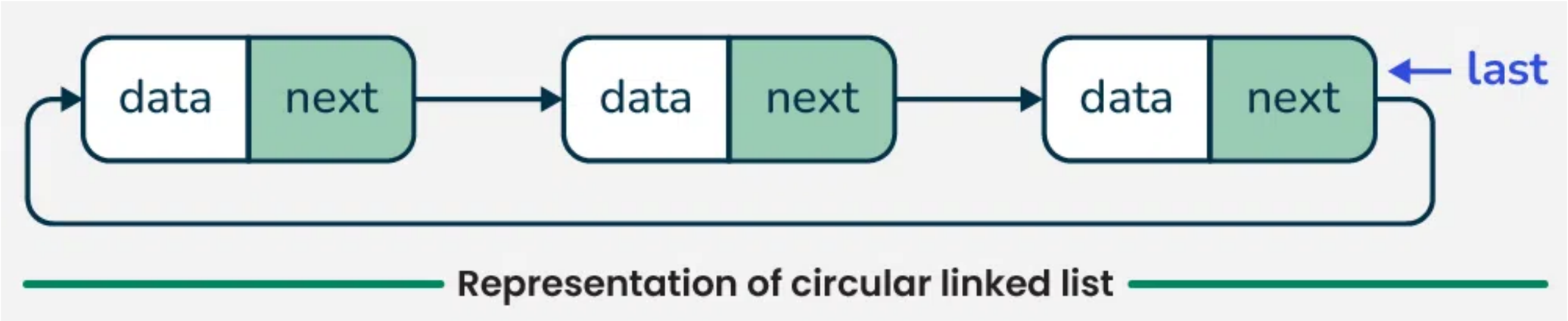
Basic Operations

- Initialization
- Insertion
- Search
- Delete
- Reverse
- ...

Circular Linked List

Definition

- All the nodes are connected to form a circle

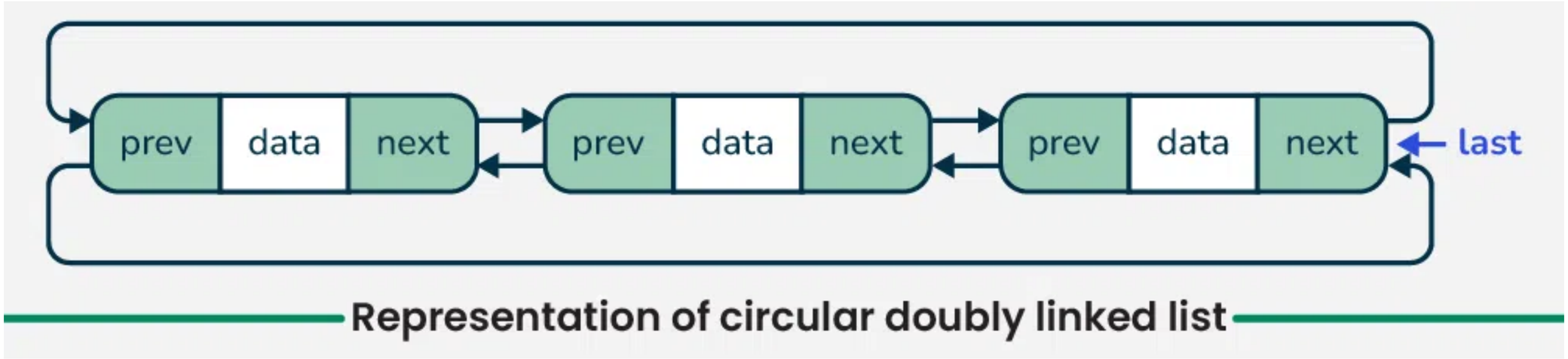


Representation of circular linked list

Circular Linked List

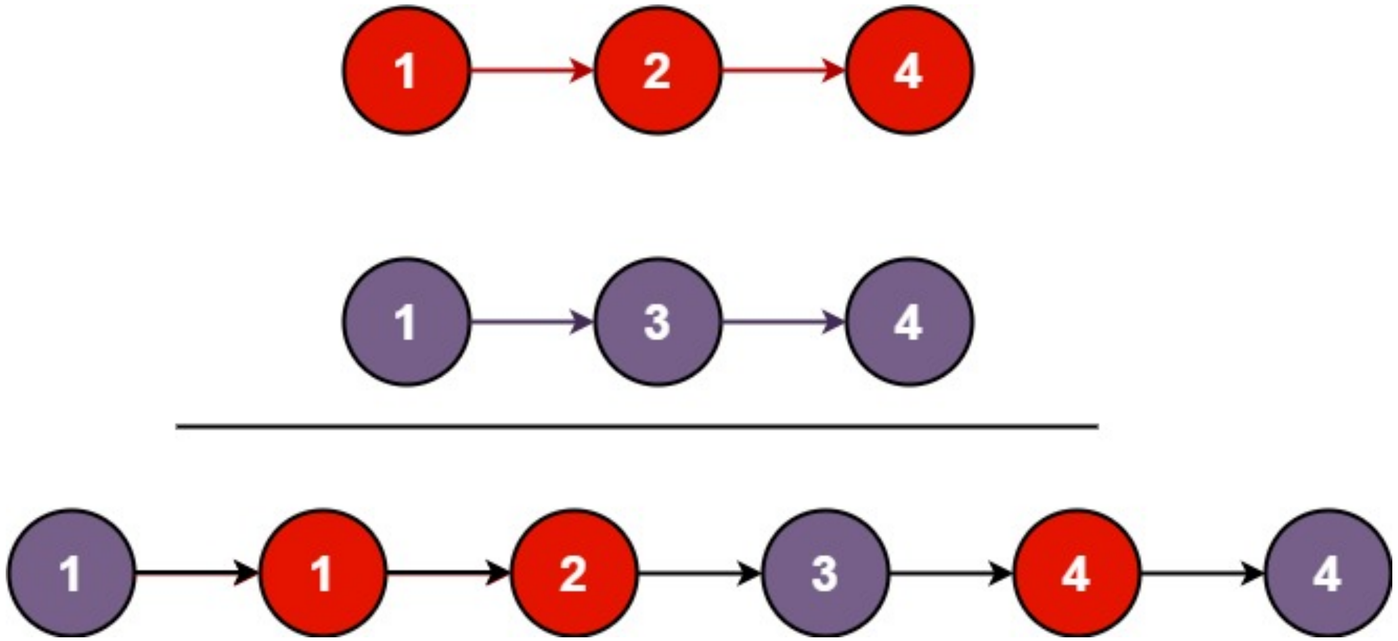
Definition

- All the nodes are connected to form a circle



Representation of circular doubly linked list

Example



- You are given the heads of two sorted linked lists *list1* and *list2*.
- Merge the two lists into one sorted list.
- The list should be made by splicing together the nodes of the first two lists.