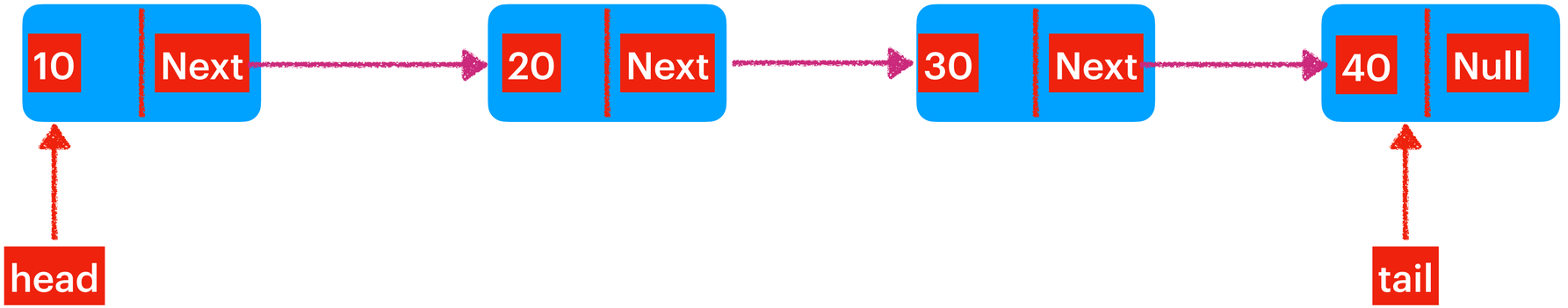
Advantages of ArrayList

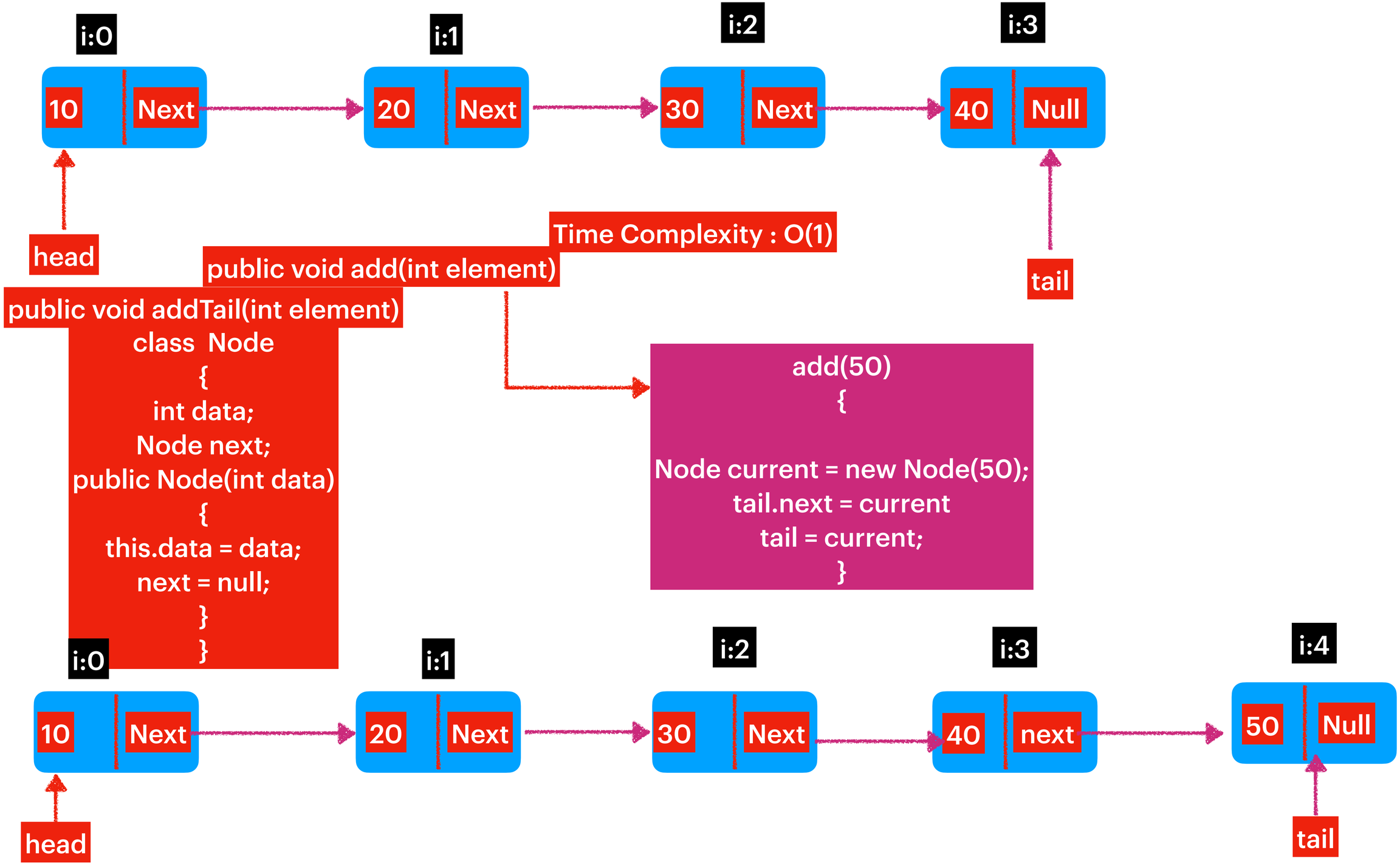
- Index based get/replace is : O(1)
- Dynamic in size

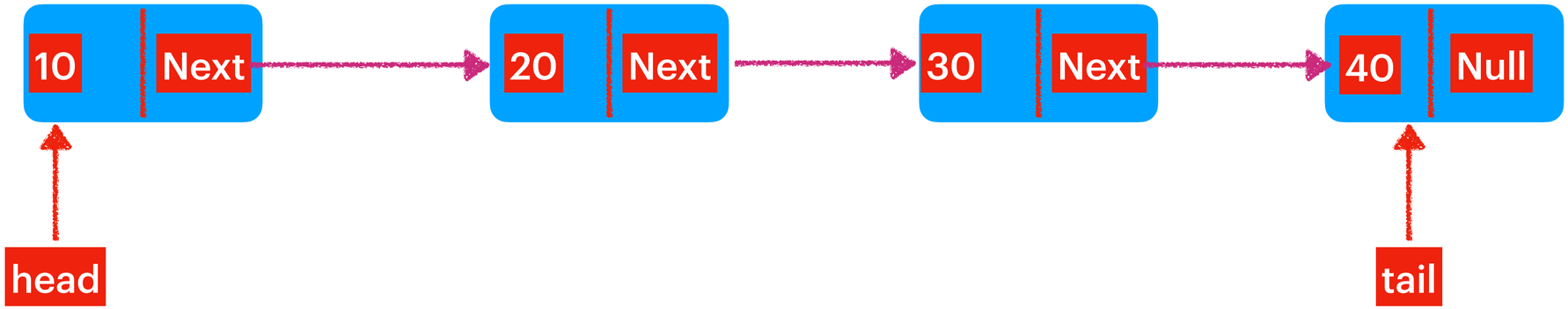
Drawbacks of ArrayList

- Delete Operation based on index takes : O(n) Time + O(n) shifts

Design LinkedList




i:0
i:1
i:2
i:3
10
Next
20
Next
30
Next
40
Null
head
tail
Time Complexity : O(1)
public void add(int element)
public void addTail(int element)
class Node
{
int data;
Node next;
public Node(int data)
{
this.data = data;
next = null;
}
}
add(50)
{
Node current = new Node(50);
tail.next = current
tail = current;
}
i:0
i:1
i:2
i:3
i:4
10
Next
20
Next
30
Next
40
next
50
Null
head
tail

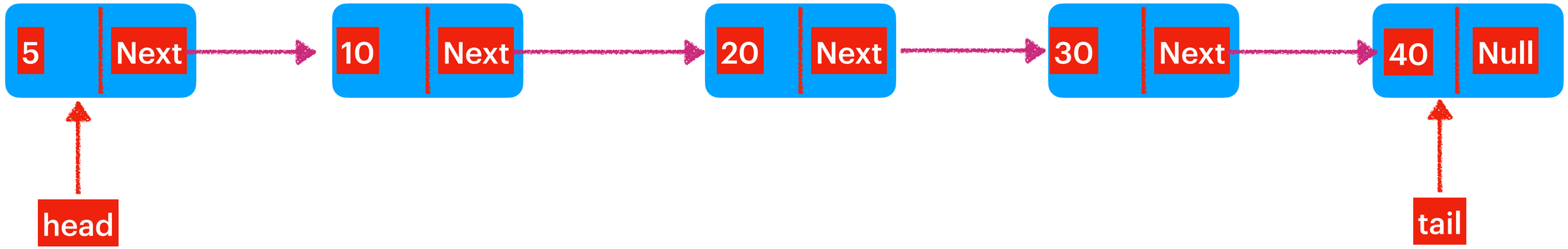


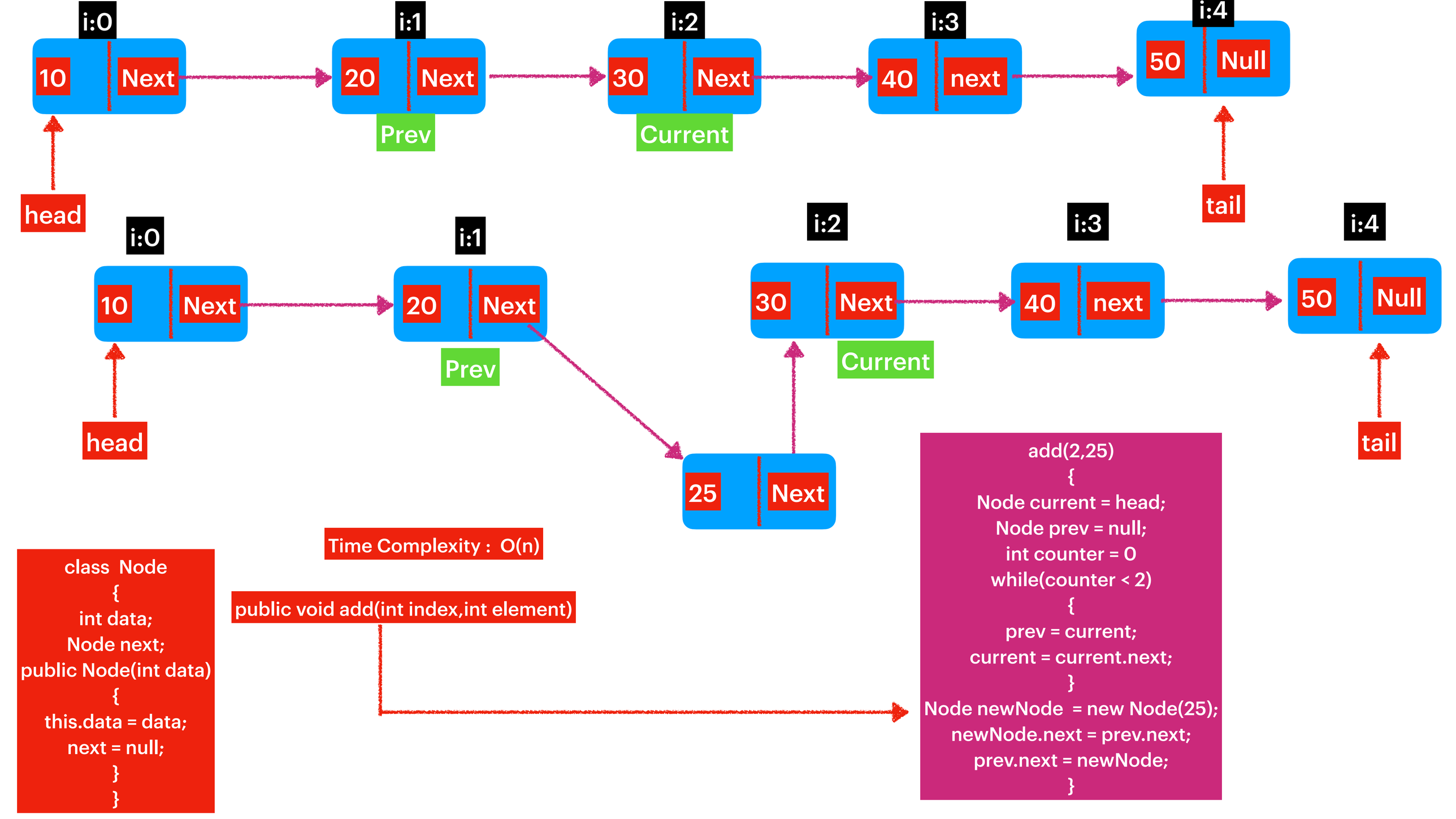
addHead(5)

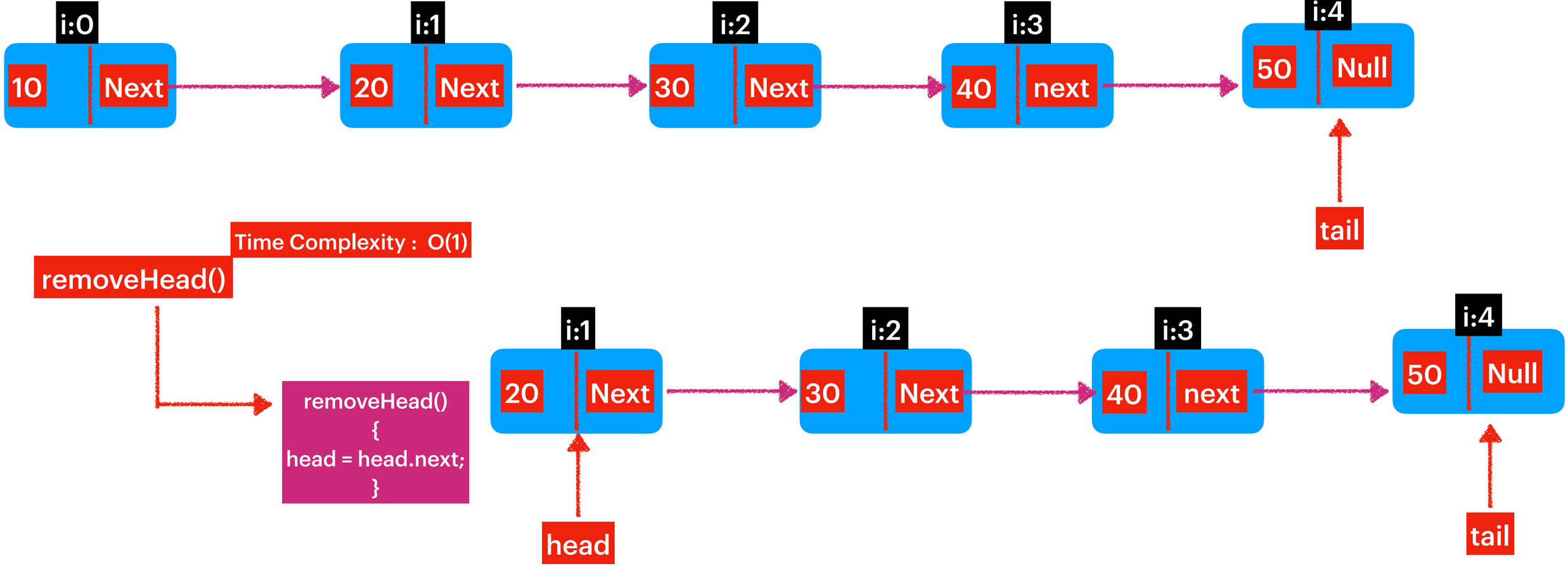
Time Complexity : O(1)

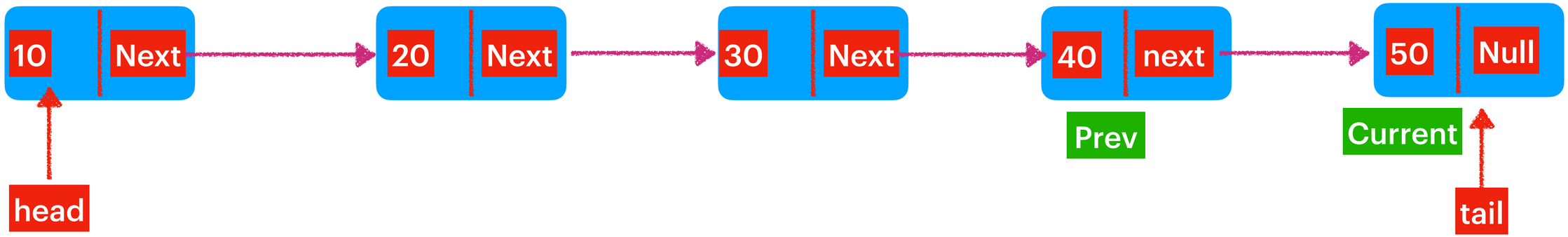
```
Node current = new Node(5)
current.next = head;
head = current;
```

```
class  Node
{
int data;
Node next;
public Node(int data)
{
this.data = data;
next = null;
}
}
```




i:0
10
Next
i:1
20
Next
Prev
i:2
30
Next
Current
i:3
40
next
i:4
50
Null
head
tail
i:0
10
Next
i:1
20
Next
Prev
i:2
30
Next
Current
i:3
40
next
i:4
50
Null
head
tail
25
Next
Time Complexity : O(n)
public void add(int index,int element)
class Node
{
int data;
Node next;
public Node(int data)
{
this.data = data;
next = null;
}
}
add(2,25)
{
Node current = head;
Node prev = null;
int counter = 0
while(counter < 2)
{
prev = current;
current = current.next;
}
Node newNode = new Node(25);
newNode.next = prev.next;
prev.next = newNode;
}

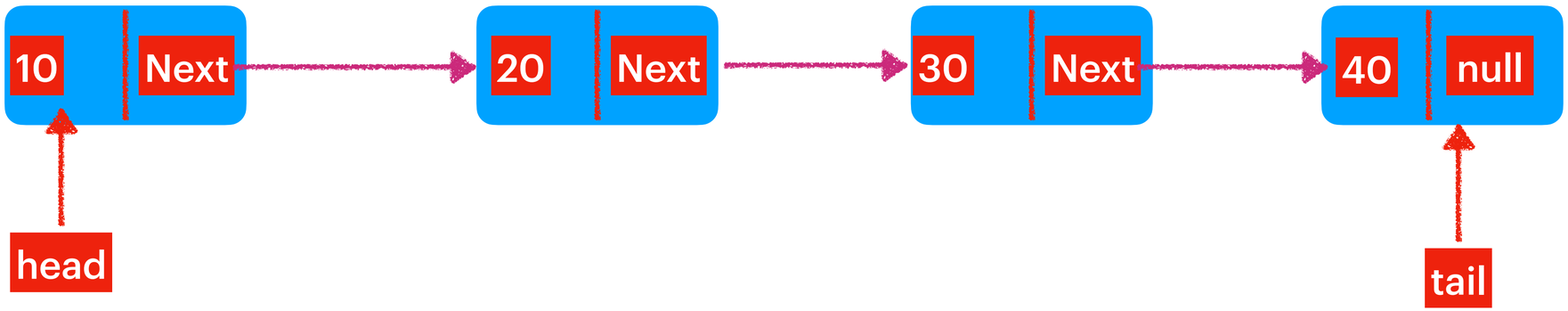

i:0
10
Next
i:1
20
Next
i:2
30
Next
i:3
40
next
i:4
50
Null
tail
Time Complexity : O(1)
removeHead()
removeHead()
{
head = head.next;
}
i:1
20
Next
i:2
30
Next
i:3
40
next
i:4
50
Null
head
tail

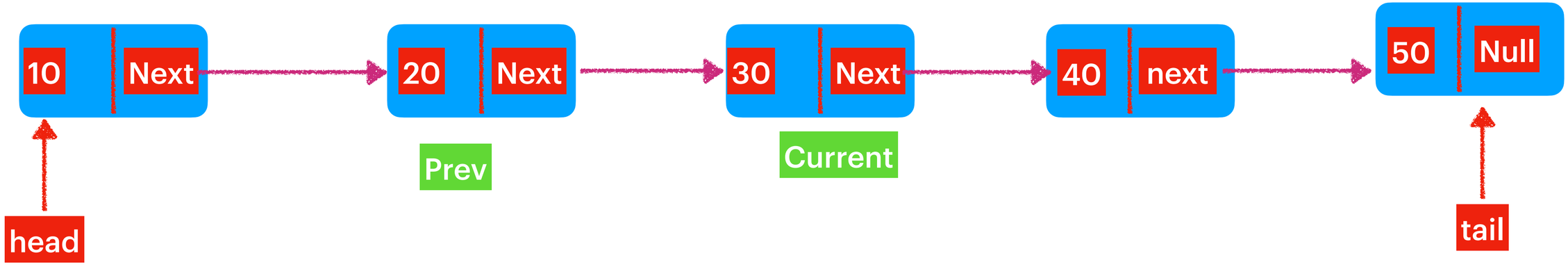


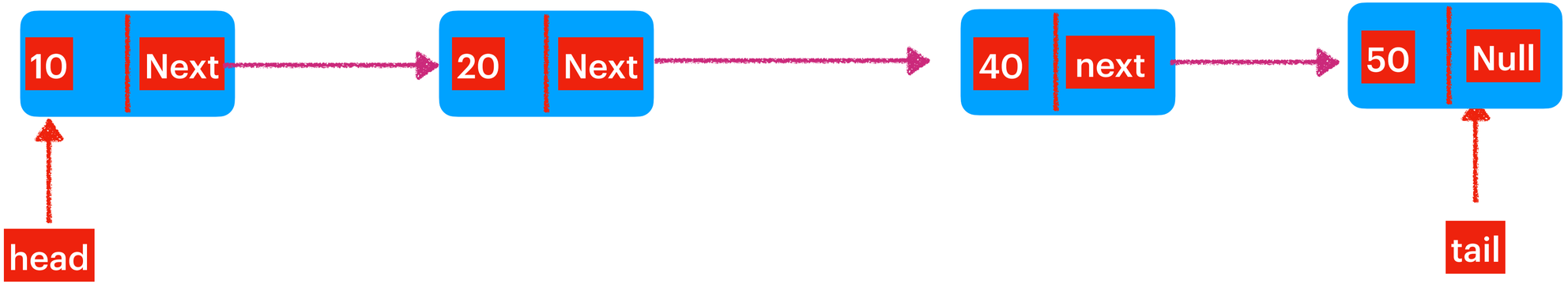
Time Complexity : O(n)

removeTail()

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







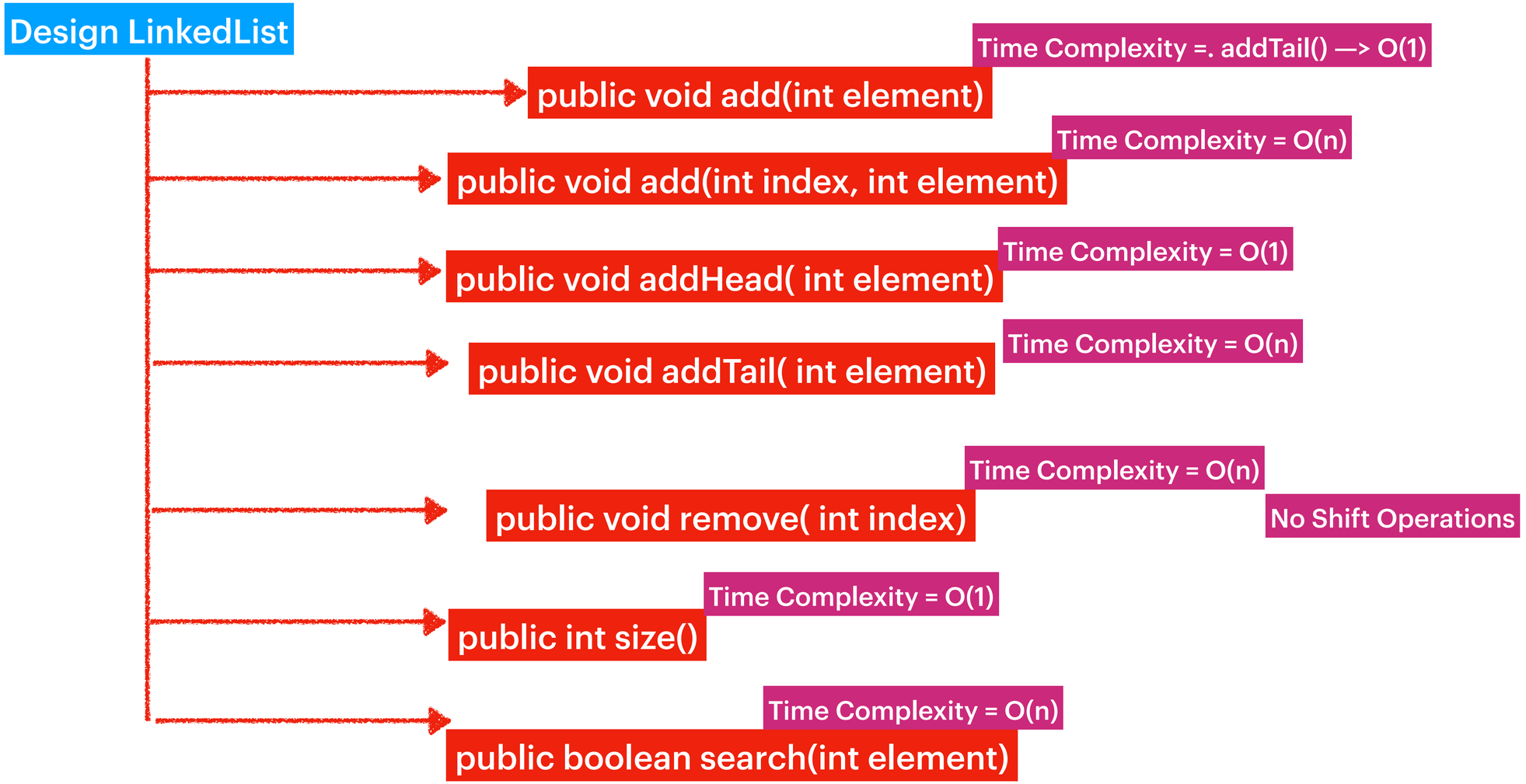

```
class  Node
{
int data;
Node next;
public Node(int data)
{
this.data = data;
next = null;
}
}
```

Time Complexity : O(n)

```
public void remove(int index)
```

```
remove(2)
{
Node current = head;
Node prev = null;
int counter = 0
while(counter < 2)
{
prev = current;
current = current.next;
}
prev.next = current.next;
current.next = null; // Helps GC
}
```

30


Design LinkedList
Time Complexity =. addTail() —> O(1)
public void add(int element)
Time Complexity = O(n)
public void add(int index, int element)
Time Complexity = O(1)
public void addHead(int element)
Time Complexity = O(n)
public void addTail(int element)
Time Complexity = O(n)
public void remove(int index)
No Shift Operations
Time Complexity = O(1)
public int size()
Time Complexity = O(n)
public boolean search(int element)