

## Case 1:

Deletion in a Linked List

Cases:

- 1 > Deleting the first node
- 2 > Deleting a node in between
- 3 > Delete the last node
- 4 > Delete a node with a given value

Case 1: Deleting the first node

struct Node \* ptr = head;

head = head → next;

Deletion in a Linked List

Cases:

- 1 > Deleting the first node
- 2 > Deleting a node in between
- 3 > Delete the last node
- 4 > Delete a node with a given value

Case 1: Deleting the first node

struct Node \* ptr = head;

head = head → next;

free(ptr);

Deletion in a Linked List

Cases:

- 1 > Deleting the first node
- 2 > Deleting a node in between
- 3 > Delete the last node
- 4 > Delete a node with a given value

Case 1: Deleting the first node

```

struct Node * ptr = head;
head = head->next;
free(ptr);

```

Deletion in a Linked List

Cases:

- 1 > Deleting the first node  $\rightarrow O(1)$
- 2 > Deleting a node in between
- 3 > Delete the last node
- 4 > Delete a node with a given value

Time complexity

Case 1: Deleting the first node


```

struct Node * ptr = head;
head = head->next;
free(ptr);

```

## Case 2:

Deletion in a Linked List



Cases:

1. Deleting the first node  $\rightarrow O(1)$
2. Deleting a node in between
3. Deleting the last node
4. Deleting a node with a given value

Case 1: Deleting the first node

```
struct Node * ptr = head;
head = head->next;
free(ptr);
```

Case 2: Deleting a node in between

```
ind = 2;
struct Node * p = head;
while (~) {
    p = p->next;
}
```

Case 2: Deleting a node in between

```
ind = 2;
struct Node * p = head;
while (~) {
    p = p->next;
}
struct Node * v = p->next;
p->next = v->next;
free(v);
```

Deletion in a Linked List

Cases:

1. Deleting the first node  $\rightarrow O(1)$
2. Deleting a node in between
3. Delete the last node
4. Delete a node with a given value

Case 1: Deleting the first node

```

struct Node * ptr = head;
head = head->next;
free(ptr);

```

Case 2: Deleting a node in between

```

ind = 2;
struct Node * p = head;
while (~) {
    p = p->next;
}
struct Node * q = p->next;

```

Deletion in a Linked List

Cases:

1. Deleting the first node  $\rightarrow O(1)$
2. Deleting a node in between
3. Delete the last node
4. Delete a node with a given value

Case 1: Deleting the first node

```

struct Node * ptr = head;
head = head->next;
free(ptr);

```

Case 2: Deleting a node in between

```

ind = 2;
struct Node * p = head;
while (~) {
    p = p->next;
}
struct Node * q = p->next;

```

Case 3: Delete the last node

```

p->next = NULL;
free(q);

```

### Case 3:

Deletion in a Linked List - OneNote

Deletion in a Linked List

Head → 4 → 3 → 8 → 1 → NULL

0 1 2 3

✓ Cases:

- 1 → Deleting the first node →  $O(1)$
- 2 → Deleting a node in between
- 3 → Delete the last node
- 4 → Delete a node with a given value (first node with the)

Case 3: Delete the last node

$p \rightarrow next = NULL$   
 $free(q);$

Case 1: Deleting the first node

$struct Node * ptr = head;$   
 $head = head \rightarrow next;$   
 $free(ptr);$

Case 2: Deleting a node in between

$ind = 2;$   
 $struct Node * p = head;$   
 $while(1) \{$   
 $p = p \rightarrow next;$   
 $\}$   
 $struct Node * q = p \rightarrow next;$

#### Case 4:

Deletion in a Linked List - OneNote

Deletion in a Linked List

Head → 4 → 3 → 8 → 1 → NULL

0 1 2 3

✓ Cases:

- 1 → Deleting the first node →  $O(1)$
- 2 → Deleting a node in between
- 3 → Deleting the last node
- 4 → Deleting a node with a given value (first node with that value)

Case 1: Deleting the first node

```
struct Node *ptr = head;
head = head → next;
free(ptr);
```

Case 2: Deleting a node in between ✓

```
ind = 2;
struct Node *p = head;
while (~) {
    p = p → next;
}
struct Node *q = p → next;
```

Case 3: Delete the last node

```
p → next = NULL;
free(q);
```