Introduction to Linked Lists :→

It is a data structure containing entities called "Nodes". The nodes can be of many types:

* | Data | Next | → only linked to next node.
* | Prev | Data | Next | → linked to next & prev nodes.

Based on this, there are three types.

* Singly linked list
* Doubly linked list
* Circular linked list → Singly / Doubly
* XOR linked list

head
| D | N | → | D | N | → | D | N | → null
Node tail

h → t → Singly Linked List
→ Only forward traversal

① → ② → ③ → null

Important Methods :→

Insert:
(i) Insert At Head / Add First
(ii) Insert At Tail / Add Last
(iii) Insert After Specific

Delete:
(iv) Delete First / head
(v) Delete Last / Tail
(vi) Delete Target

curr
A → B → C → D → null (stop)
head tail

```
while (curr != null) {
    print(value);
    c = c → next;
}
```

Add First :→

newNode (h) X → A → B → C → null
(head)

① nN.next = head
h = nN

A → B → C → D → E → null
head last last last last
New Node last stop

```
Node last = head;
while (last.next != null) {
    last = last.next;
}
last.next = newNode;
```

(iv)

A → null
head this.
delete( )
free( )

```
head = head.next;
```

① → ② → ③ → null
garbage head h
JVM
JVM handles garbage

```
head = this.head.next;
size--;
```

this → calling object

Delete Last Node in a List with multiple nodes.

head curr curr
A → B → C → D → null
curr last last last last Node
stop

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

Doubly Linked List :→

Both Forward & Backward Traversals

null ← | P | D | N | ↔ | P | D | N | ↔ | P | D | N | → null
head tail

head.prev = null
tail.next = null

Node structure ←

```
class Node {
    Node next;
    Node prev;
    int data;
}
```

* LeetCode Questions :→

(i) Find the Middle Node of a Linked List
(ii) Delete the Middle Node of a Linked List
(iii) Merge two Sorted Linked Lists.
(iv) Reverse A Linked List
(v) Intersection Point of two Linked Lists
(vi) Floyd's Cycle Finding Algorithm (Hare & Tortoise)

slow slow slow
① → ② → ③ → ④ → ⑤
fast fast fast

```
while (fast.next != null) {
    slow = slow.next;
    fast = fast.next.next;
}
```

slow slow slow slow
① → ② → ③ → ④ → ⑤ → ⑥ → null
fast head fast fast fast

```
slow = head
fast = head
while (fast.next != null)
    slow = slow.next;
    fast = fast.next.next;
}
```

① → ② → ③ → ④ → ⑤ → null
temp temp temp
no of nodes = 5
m 5/2 = 2, c = 0 (1, 2)

```
while (c < m) {
    temp = temp.next
    c++;
}
```

stop
0 < 2 1 < 2 2 < 2
return temp;

① → ② → ③ → ④ → ⑤ → ⑥ → null
temp temp temp temp
no of nodes = 6
m 6/2 = 3, c = 0

```
while (c < m) {
    temp = temp.next
    c++;
}
```

c = 0, 1, 2, 3
0 < 3 1 < 3 2 < 3 3 < 3
return temp;

temp curr curr.next
① → ② → ③ → ④ → ⑤ → null
temp ↓ null curr ↓ head JVM
m = 2
5/2 = 2

```
c=0 while (count < m) {
    temp = curr;
    curr = curr.next;
    c++;
}
temp.next = curr.next;
head;
return head;
```

Floyd's Hare & Tortoise Algorithm to find cycle in a Linked List:

slow slow slow slow
① → ② → ③ → ④
fast fast fast fast

```
while (fast && fast.next != null)
    slow = slow.next;
    fast = fast.next.next;
    if (slow == fast) return true;
return false;
```

* Given a linked list, check whether it is a "Palindrome" or not.

① → ③ → ③ → ① → null
h False t

1st → 1 3 2 1
2nd → 1 == 1, 3 == 3, 3 == 3, 1 == 1
true, false

* Stack LIFO (top): 1, 3, 3, 1

Iterator? → Interface

Use the iterator to traverse an ArrayList & remove elements on certain condition.

(java.util) Collections.

5 4 le Coders Arcade