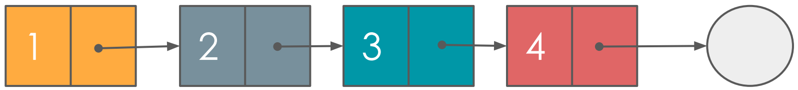
Linked List I



Lecture Flow

- Pre-requisites
- Problem definitions and applications
- Types of linked list
- Different approaches
- Variants
- Recognizing in questions
- Common Pitfalls
- Practice Questions

Pre-requisites

1. Arrays
2. Pointers

A2SV Africa To Silicon Valley

Definitions

Linked List is a linear data structure that stores value and grows dynamically

Linked List consists of nodes where each node contains a data field and a reference(link) to the next node in the list

Africa To Silicon Valley

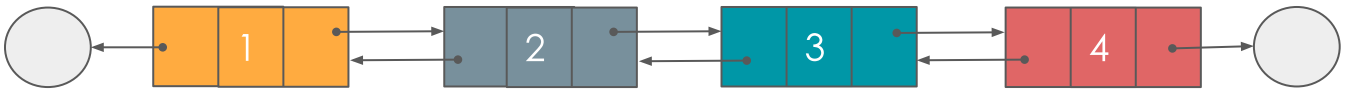
Node

- stores the value and the reference to the next node.
- The simplest linked list example

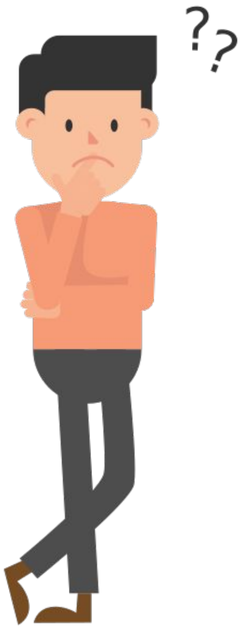
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None # Type: Node
```

Singly Linked List is when nodes have only next's node reference.

Doubly Linked List is when nodes have both previous and next node reference.



Why do we need linked list when we have arrays?



Why Linked List when you have Arrays?

- Arrays by default don’t grow dynamically
- Inserting in the middle of an array is costly
- Removing elements from the middle of array is costly

Arrays vs Linked List

Array	Linked List
Fixed size	Dynamic size
Insertions and Deletions are inefficient	Insertions and Deletions are efficient
Random access	No random access
Possible waste of memory	No waste of memory
Sequential access is faster	Sequential access is slow

Common Operations on Linked List

Traversing a Linked List

- Start with the head of the list. Access the content of the head node if it is not null
- Go to the next node(if exists) and access the node information
- Continue until no more nodes (that is, you have reached the last node)

Problem

Write a function that returns an array representation of a given linked list.

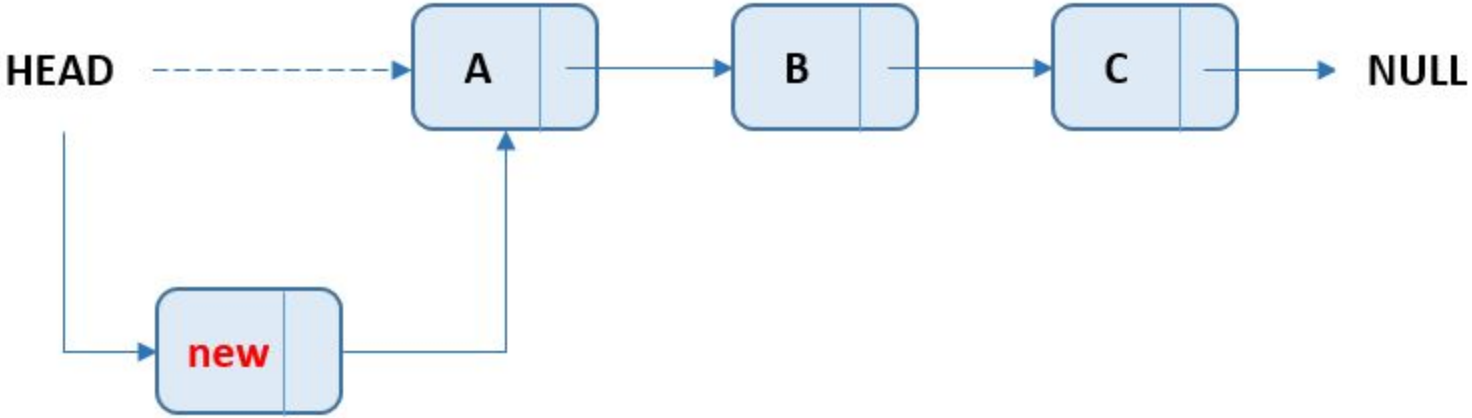
```
a = Node(1)

b = Node(2)

c = Node(3)

a.next = b

b.next = c
```

[1, 2, 3]

Inserting a node in linked list

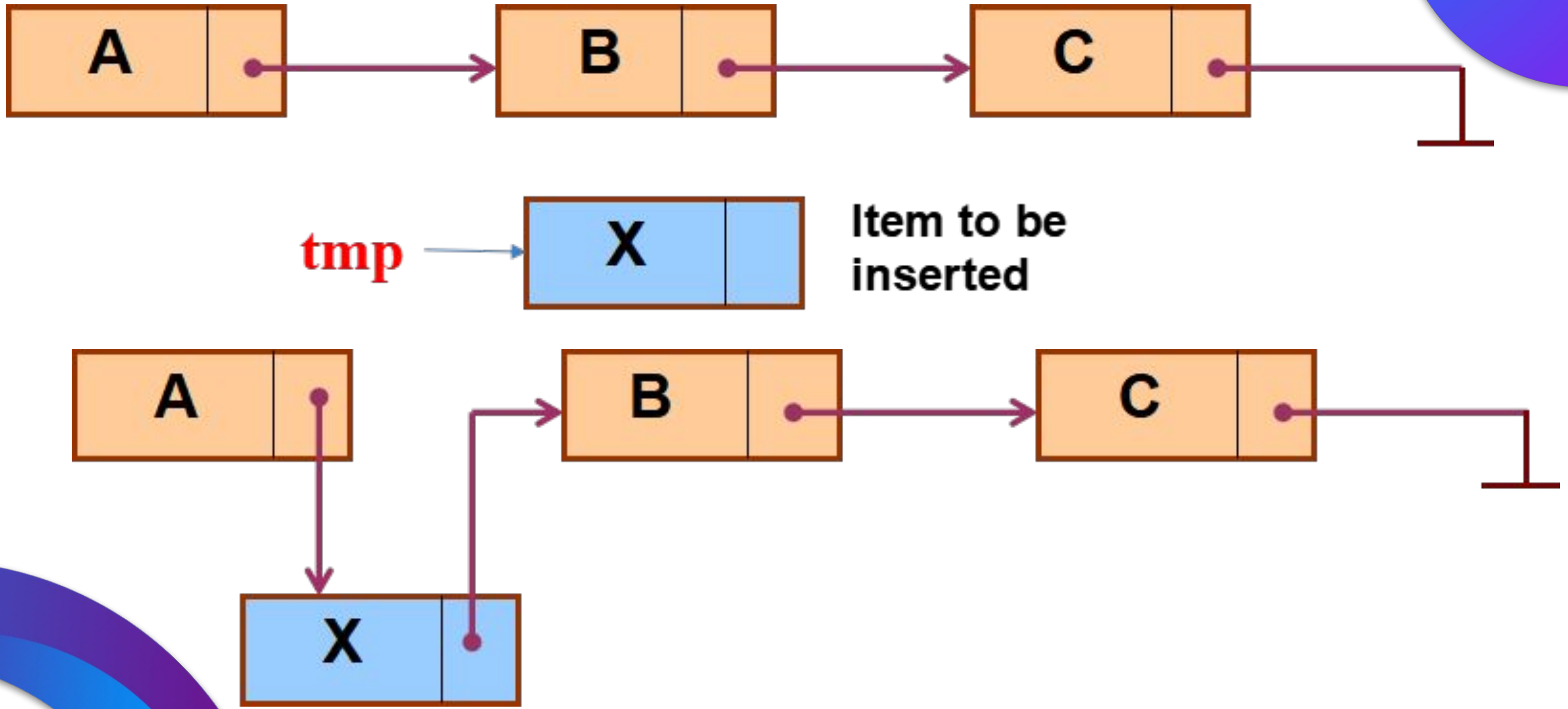
Insert at the beginning

- If list is empty
  - make new node the head of the list
- Otherwise
  - connect new node to the current head
  - make new node the head of the list.



Insert at any position

- Find the insert position and the previous node
- And then make the next of new node as the next of previous node
- Finally, make the next of the previous node the new node


A
B
C
tmp
X
Item to be inserted
A
B
C
X

Can we merge the two insertions into one function?

How?

Yes, we can use Dummy Node before the head.

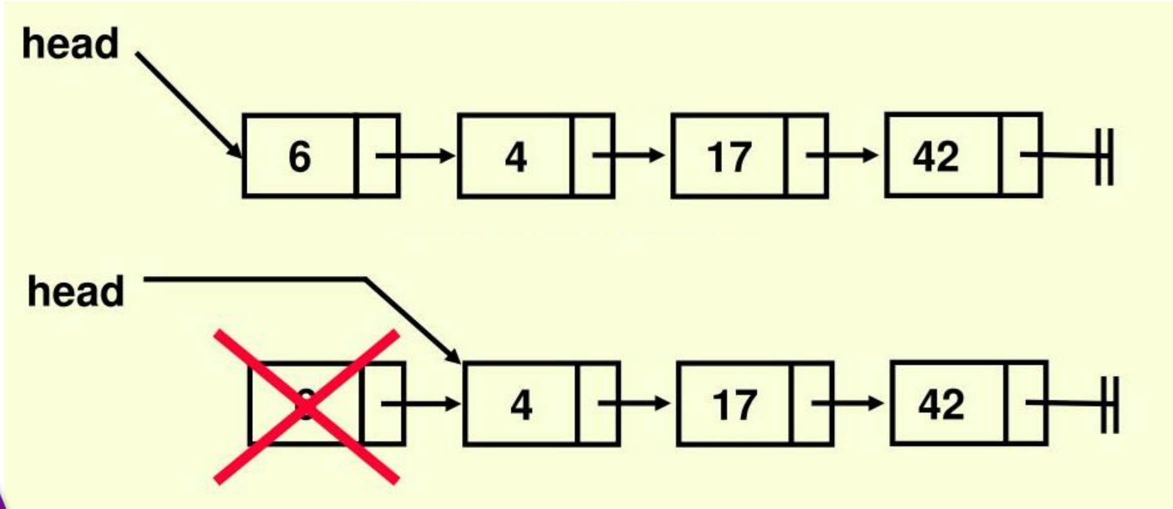
Dummy Node

- is a node that points to the head of a linked list which will be discarded at the end
- When to use a Dummy Node?
    - if you are potentially modifying the head of linked list, use dummy node

Delete a node from the linked list

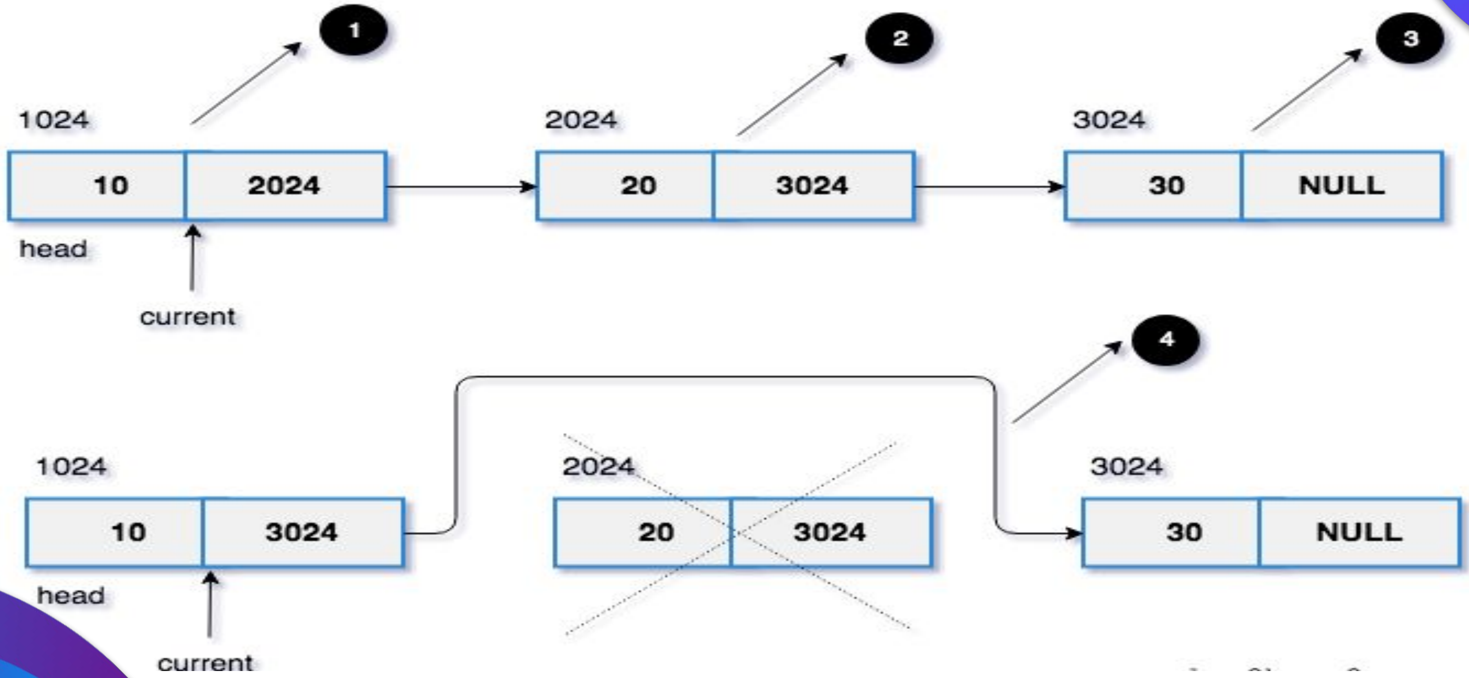
Delete a node at the beginning

- Make the second node as head
- Discard the memory allocated for the first node.



Delete a node at any position

- Find a match the node to be deleted
- Get the previous node
- Make the previous node next point to the next of the deleted node


1
2
3
1024
10
2024
head
current
2024
20
3024
3024
30
NULL
4
1024
10
3024
head
current
2024
20
3024
3024
30
NULL

Can we avoid using two approaches when we are deleting nodes? How?

Yes again! We can use Dummy Node.

A2SV
Africa To Silicon Valley

Linked List Class

- The LinkedList class serves as the container for the nodes.
- The __init__ method initializes an empty linked list with a head pointing to None.

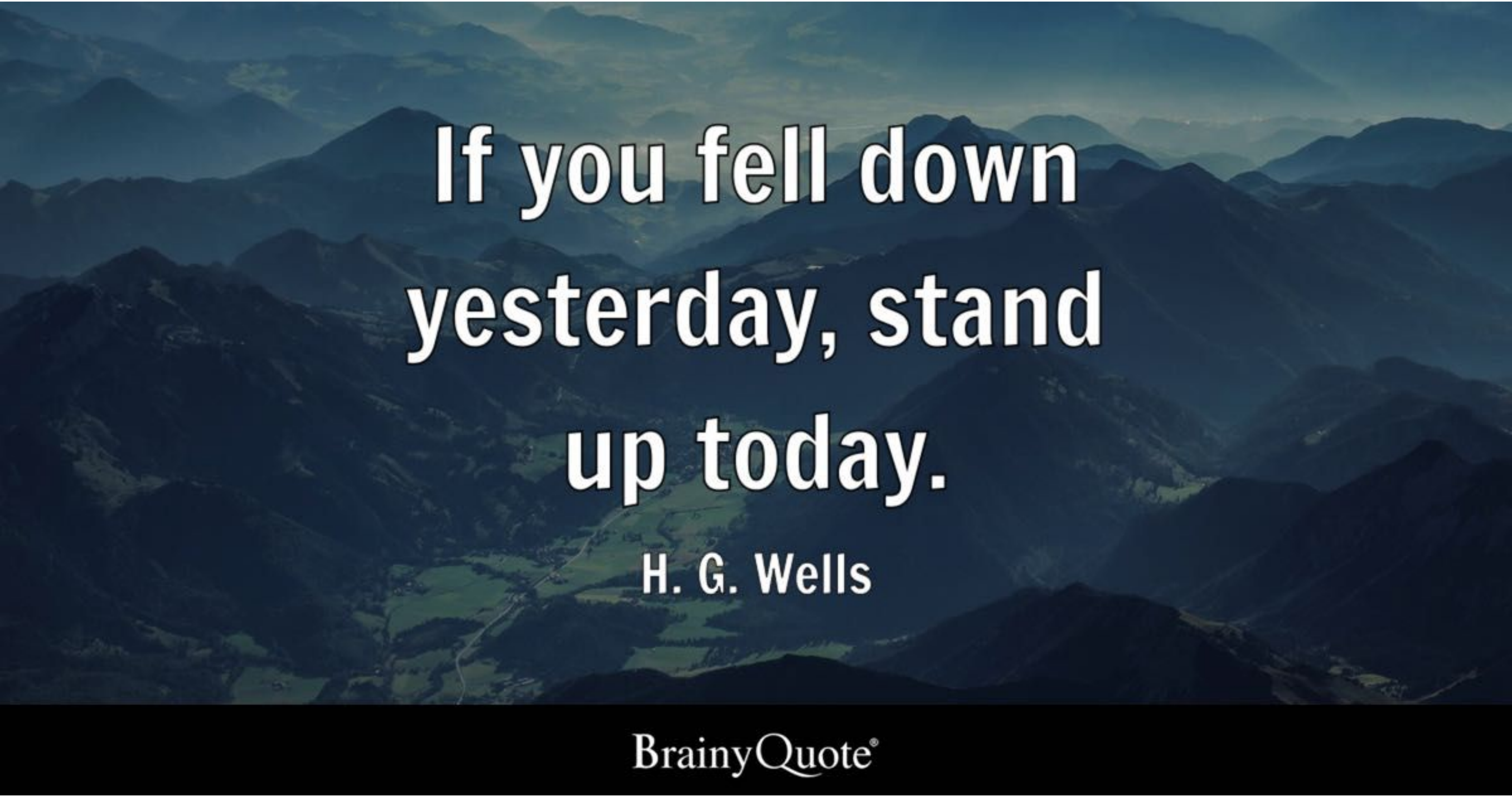
```python
class LinkedList:
    def __init__(self):
        self.head = None
```

Pair Programming

Design Linked List(implement deleteAtIndex)

Resources

- Leetcode Explore Card: has excellent track path with good explanations
- Leetcode Solution (Find the Duplicate Number) : has good explanation about Floyd's cycle detection algorithm with good simulation
- Elements of Programming Interview book: has a very good Linked List Problems set


If you fell down yesterday, stand up today.
H. G. Wells
BrainyQuote®