Linked Lists Intro

What is a linked list ?
List created by linking elements called as nodes !!

In [ ]:

Node structure and self referential struct

```cpp
// C++
struct Node {
    int data;
    Node *next;
};

// Decimal 0-9  - 10
// Hexa 0-9 ABCDEF - 16
```

Dynamic memory allocation

```cpp
int main() {
    Node *n1 = new Node();
    n1->data = 10;
    n1->next = NULL;

    Node *n2 = new Node();
    n2->data = 20;
    n2->next = n1;
}
```

Complexity Analysis

In [ ]:
```
- Insert at front: O(1)
- Insert at end: O(1)
- Delete at end: O(1)
- Delete at end: O(1)
- Delete from middle:
    - Not given the refrence to the node to be delete O(n)
    - Given the refrence to the node to be delete O(1)
```

In [ ]:

JAVA

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

    public Node(int data, Node next) {
        this.data = data;
        this.next = next;
    }
}
Node n1 = new Node(10, null);
Node n2 = new Node(20, n1);
```

Python

```
class Node:
    def __init__(self, data: int, next: Node):
        self.data = data
        self.next = next

n1 = Node(10, None)
n2 = Node(20, n1)
```

In [ ]:

Question
https://leetcode.com/problems/delete-node-in-a-linked-list/ (https://leetcode.com/problems/delete-node-in-a-linked-list/)

In [ ]:
```python
class Solution:
    def deleteNode(self, node):
        """
        :type node: ListNode
        :rtype: void Do not return anything, modify node in-place instead.
        """
        node.val = node.next.val
        node.next = node.next.next
```

In [ ]:
```java
public class Solution {
    public void deleteNode(ListNode node) {

        node.val = node.next.val;

        node.next = node.next.next;
    }
}
```

In [ ]:
```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
    public void deleteNode(ListNode node) {
        node.val = node.next.val;
        node.next = node.next.next;
    }
}
```

In [ ]:
```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
aclass Solution {
    public void deleteNode(ListNode node) {
        node.val=node.next.val;
        node.next=node.next.next;
    }
}
```

In [ ]:
```
```

Question

https://leetcode.com/problems/linked-list-cycle/description/ (https://leetcode.com/problems/linked-list-cycle/description/)

In [ ]:
```
Brute force solution
Using Hashmap
#  TC: O(n)
#  SC: O(n)
```

In [ ]:
```
#  TC: O(n)
#  SC : O(1)

/**
 * Definition for singly-linked list.
 * class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) {
 *         val = x;
 *         next = null;
 *     }
 * }
 */
public class Solution {
    public boolean hasCycle(ListNode head) {
        if(head == null) return false;

        ListNode slow = head;
        ListNode fast = head;

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

            if(slow == fast){
                return true;
            }
        }
        return false;
    }
}
```

In [ ]:
```
class Solution:
    def hasCycle(self, head: Optional[ListNode]) -> bool:
        curr = head
        while curr:
            if curr.next == head:
                return True
            curr.next, curr = head, curr.next
        return False
```

In [ ]:

Question
https://leetcode.com/problems/linked-list-cycle-ii/ (https://leetcode.com/problems/linked-list-cycle-ii/)
https://stackoverflow.com/questions/2936213/how-does-finding-a-cycle-start-node-in-a-cycle-linked-list-work/36214925#36214925 (https://stackoverflow.com/questions/2936213/how-does-finding-a-cycle-start-node-in-a-cycle-linked-list-work/36214925#36214925)

In [ ]:

In [ ]:

Question
https://leetcode.com/problems/intersection-of-two-linked-lists/ (https://leetcode.com/problems/intersection-of-two-linked-lists/)

Brute Force

Use a hash map
TC: O(m+n)
SC: O(max(m,n))

In [ ]:
```
# optimal
# TC: O(m+n)
# SC: O(1)

int getLen(ListNode  *head) {
    int len = 0;

    while (head != NULL){
        len ++;
        head = head->next;
    }

    return len;
}

class Solution {
public:
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {
        int l1 =0, l2  = 0;

        l1 = getLen(headA);
        l2 = getLen(headB);

        // a1
        // b2
        ListNode *s=headA, *l=headB;
        if (l1 > l2) {
            l = headA;
            s = headB;
        }


        for(int i = abs(l1-l2); i >0; i-- ) {
            l = l->next;
        }

        while(s != NULL) {
            if (s == l) {
                return s;
            }
            s = s->next;
            l = l->next;
        }

        return NULL;
    }
};
```

In [ ]:
```

```

Question

https://leetcode.com/problems/reverse-linked-list/ (https://leetcode.com/problems/reverse-linked-list/)

In [ ]:

In [ ]:

Question
https://leetcode.com/problems/merge-two-sorted-lists/ (https://leetcode.com/problems/merge-two-sorted-lists/)

In [ ]:

In [ ]:

Question

https://leetcode.com/problems/reverse-nodes-in-k-group/ (https://leetcode.com/problems/reverse-nodes-in-k-group/)

In [ ]:

In [ ]: