Linked Lists-3

In [ ]:
```
1
```

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

Solution-1
TC: O(N)
SC: O(N)

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    bool hasCycle(ListNode *head) {
        set<ListNode*> visited;
        while(head != NULL) {
            // Check if the pointer 'head' already exists in the set
            if (visited.find(head) != visited.end()) return true;

            visited.insert(head);
            head = head->next;
        }
        return false;
    }
};
```

Solution-2: Two pointer
TC: O(N)
SC: O(1)

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    bool hasCycle(ListNode *head) {

        ListNode *slow=head, *fast=head;

        while(fast != NULL && fast->next != NULL) {

            slow = slow->next;
            fast = fast->next->next;

            if(slow == fast) return true;
```

Type *Markdown* and LaTeX: α^2

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)

Solution-1: Brute Force TC: O(N)
SC: O(N)

```cpp
    /**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        set<ListNode*> visited;
        while(head != NULL) {
            // Check if the pointer 'head' already exists in the set
            if (visited.find(head) != visited.end()) return head;

            visited.insert(head);
            head = head->next;
        }
        return NULL;
    }
};
```

Solution-2: Fast, Slow
TC: O(N) SC: O(1)

```
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {

        if (head == NULL || head->next == NULL) return NULL;

        ListNode *slow=head, *fast=head;

        while(fast != NULL && fast->next != NULL) {

            slow = slow->next;
```

In [ ]:
```
1
```

In [ ]:
```
1
```

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

Solution-1: Brute Force
TC: O(m + n)
SC: O(max(m,n))
C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
```

Solution-2

TC: 2*O(m + n) = O(m + n)

SC: O(1)

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {
        set<ListNode*> visited;

        int lenA=0, lenB=0;

        // Find lenght of both Linked Lists
        ListNode *temp;
        temp = headA;
        while(temp != NULL) {
            lenA += 1;
            temp = temp->next;
        }
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1 a ^ a -> 0
2 a ^ b ^ a -> b
```