

Arrays, Strings & Linked Lists Lecture 7

Tuesday, 30 July 2024 6:05 AM

obj1 → obj2 → obj3 → ...

struct Node

int data;

Node * next;

Node * head;

head → data = x;

head → next = t;

(*head).data

class Node:

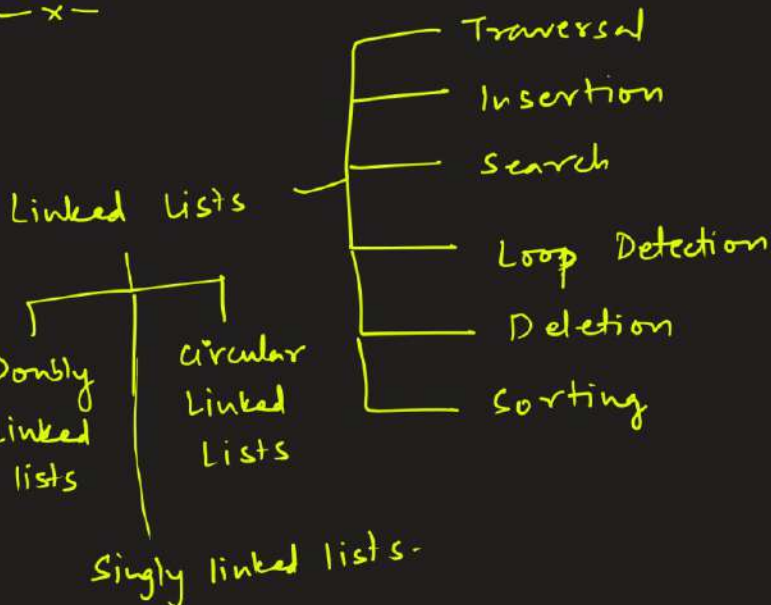
def __init__(self, data):

self.data = data

✓ self.next = None

{ new_node = Node(x)
new_node.next = head;
head = new_node

-x-



<https://www.geeksforgeeks.org/problems/print-linked-list-elements/1>

class Solution:

Function to display the elements of a linked list

def display(self, head):

#code here

t = head

while t:

print(t.data, end=' ')

t = t.next

print("")

class Solution {

public:

// Function to display the elements of a linked list

void display(Node *head) {

// your code goes here

Node* t = head;

while(t) {

cout << t->data << " ";

t = t->next;

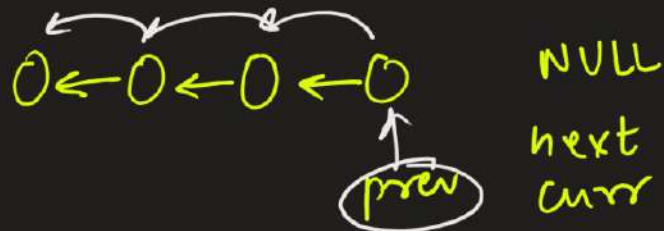
}

cout << endl;

}

};

<https://leetcode.com/problems/reverse-linked-list>



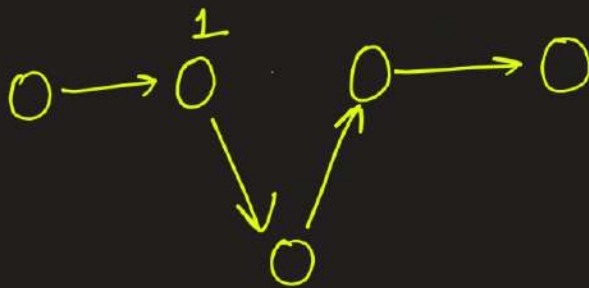
$T = O(n)$

$S = O(1)$

```
next = curr->next;
curr->next = prev;
prev = curr;
curr = next;
```

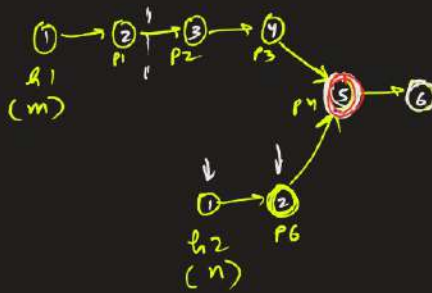
```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = NULL, *curr = head, *next = NULL;
        while(curr) {
            next = curr->next;
            curr->next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }
};
```

<https://www.hackerrank.com/challenges/insert-a-node-at-a-specific-position-in-a-linked-list/problem>



```
SinglyLinkedListNode* insertNodeAtPosition(SinglyLinkedListNode* llist, int data, int position) {  
    SinglyLinkedListNode *newNode = new SinglyLinkedListNode(data);  
    if(position == 0){  
        newNode->next = llist;  
        return newNode;  
    }  
    SinglyLinkedListNode* t = llist;  
    for(int i=0; i<position-1; i++)  
        t = t->next;  
    newNode->next = t->next;  
    t->next = newNode;  
    return llist;  
}
```

<https://www.hackerrank.com/challenges/find-the-merge-point-of-two-joined-linked-lists/problem>



Brute:- $O(mn)$ }

Hashing: $T = O(m+n)$
 $S = O(m)$

Count:- $T = O(m+n)$
 $S = O(1) \leftarrow 2 \text{ counts}$

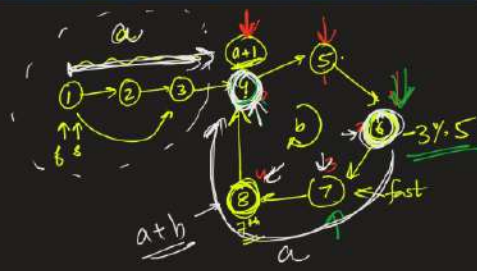
2-pointers:- $T = O(m+n)$
 $S = O(1)$

Detect loop:- $T = O(m+n)$
 $S = O(1)$ }

Count (h_1) $\rightarrow O(m) \rightarrow \frac{m}{6}$
 Count (h_2) $\rightarrow O(n) \rightarrow \frac{n}{4}$
 $|m-n| = 2$
 bigger list at this element

```

int findMergeNode(SinglyLinkedListNode* head1, SinglyLinkedListNode* head2) {
    SinglyLinkedListNode *p1 = head1, *p2 = head2;
    while(p1 != p2) {
        p1 = p1->next;
        p2 = p2->next;
        if(p1 == NULL)
            p1 = head2;
        if(p2 == NULL)
            p2 = head1;
    }
    return p1->data;
}
  
```



Total # ele = a

	pos	steps
S	a+1	a
f	a%b+a+1	a
	(2k+1-a-1)%b+a+1	

From the int. point if you move 'a' steps you land at cycle starting point

Floyd's cycle detection algorithm

k hops → fast ptr?

- 0 hops → 1
- 1 hop → 3
- 2 hops → 5
- 3 hops → 7
- 4 hops → 9

- 5 hops → 11
- 6 hops → 13
- 7 hops → 15
- k hops → 2k+1

$k = \underline{b-a}$
 $a+1 + (b-a)$
 $= \underline{b+1}$
 $\underline{a+b+1}$

S → a+1
f → a+1 + a%b

after k steps
 $\underline{a+1+k} = a+1 + (2k+a)\%b$

$k = (2k+a)\%b$

$(k+a)\%b = 0$

$\underline{k = -a\%b}$ ✓

$\underline{k = (b-a)}$

$(\underline{p+2k-a-1})\%b + a+1$

$\underline{(2k+a\%b)\%b + a+1}$

$a+1 + (2k+a)\%b$