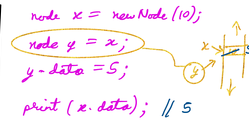
Linked List 2

Wednesday, 15 September 2021 9:05 PM

Deep Copy

```
Node x = new Node(10);
Node y = new Node(x.data);
y.data = 5;
print(x.data); // 10
```

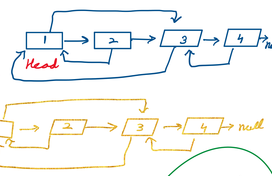
Shallow Copy

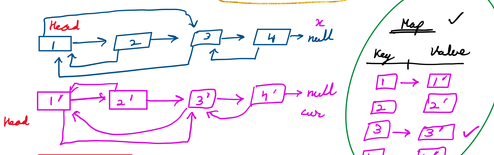


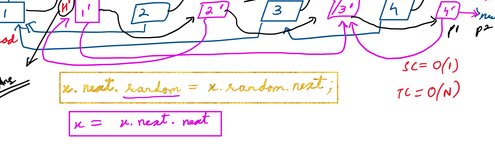
```
Node x = new Node(10);
Node y = x;
y.data = 5;
print(x.data); // 5
```

Q→ For a given Linked list, create its deep copy. (LC World)

```
struct node {
    int data;
    node * next, * random;
}
```







TC = O(N)
SC = O(N) ✓

✓ cur.random = map[x.random]

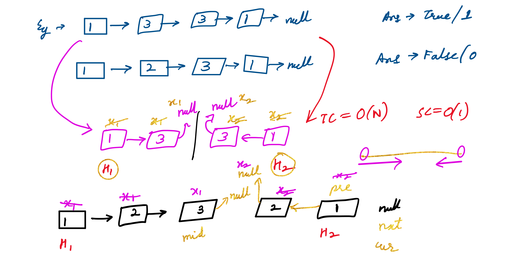
x.next.random = x.random.next;

x = x.next.next

```
p1.next = p2.next
p1 = p2
p2 = p2.next
```

SC = O(1)
TC = O(N)

Q→ Check if the given linked list is palindrome bit.

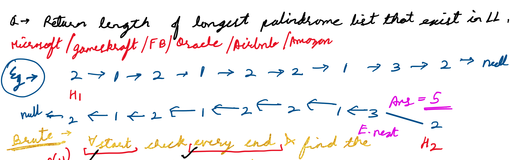


Ans → True/1

Ans → False/0

TC = O(N) SC = O(1)

Q→ Return length of longest palindrome list that exist in LL. (Microsoft / Gamecraft / FB / Oracle / Airbnb / Amazon)



Brute → ∀ start, check every end, ∴ find the ∃ next sub-linked list with max length which is palindrome.

SC = O(N)

```
S = H1;  ans = 0;  size = n;
while(S != null) {
    E = H2;  len = size;
    while(len > 0) {
        if(check(len, S, E)) {   // check if S-E is palindrome
            ans = max(ans, len);
            break;
        }
        E = E→next;  len--;
    }
    S = S→next;  size--;
}
return ans;
```

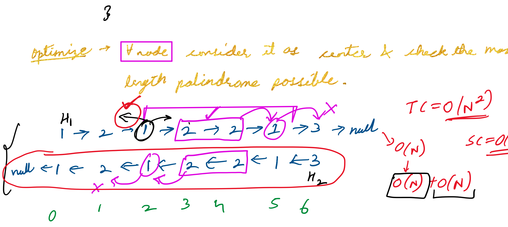
∀ start

∀ endpoints

TC = O(N^3)
SC = O(N)

```
bool check(len, p1, p2) {        // p1 → S, p2 → E
    while(len > 1 && p1.data == p2.data) {
        len -= 2;
        p1 = p1.next;
        p2 = p2.next;
    }
    return (len <= 1);
}
```

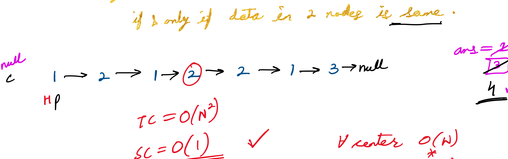
optimise → ∀ node consider it as center & check the max length palindrome possible.



TC = O(N^2)
SC = O(N) ✓

O(N) * O(N)

Some way consider every 2 nodes to be the center, if & only if data in 2 nodes is same.



ans = 4

TC = O(N^2)
SC = O(1) ✓

∀ center O(N)
Trend O(N)