N.

N ← 1 ⇄ 2 ⇄ 3 ⇄ 4 → N.

M.

curr

N ← 1 ⇄ 2 ⇄ 2 ⇄ 2 ⇄ 3 ⇄ 3 ⇄ 4 → N.

t

Off:

N ← 1 ⇄ 2 ⇄ 3 ⇄ 4 → N.

curr → next = t

if t != N (t → prev = curr)

curr = t

N ← 1 ⇄ 2 ⇄ 2 ⇄ 2 ⇄ 3 ⇄ 3 ⇄ 4 → N.

t = curr → next

N ← 1 ⇄ 2 ⇄ 3 ⇄ 4 → N

In t → Node

TC = O(n).

AS = O(1).

STACK.

LIFO → Last In First Out.

class Stack {

vector <int> arr;

public:

push() → push_back() → O(1).

pop() → pop_back() → O(1)

top() → return arr[sz-1] → O(1)

isEmpty() → O(1)

size() → O(1)

:

}

1, 2, 3, 4, 5

5
4
3
2
1

-5
4
3
2
1

class Stack {

LinkedList ll, int size;

public:

push() → Insert at Begin → O(1)

pop() → Del " " → O(1)

top() → head→data.

isEmpty() → O(1).

size() → O(1)

}

5 → 4 → 3 → 2 → 1 → N.

F

~~1 → 2 → 3 → 4~~ → 5 → N.

6
5
4
3
2
1

---

k = 1 k

[( )] → ✓

[( ]) → ✗

k k

[[ ]] → ✗

[[{ }]] → ✓

[ ] { } → ✓

k k k k

[{ }] ( )]

LIFO

{ { { [

k k l

[[ ]] → ✗

[

[

k k k k l

[[ ]]

[{ }]

[ ] { }

{ } [ ]

TC = O(n).

AS = O(n).

l

] ( ).

k k l

(( ).

✗

(

---

arr: [6, 8, 0, 1, 3]

ngl[]: 8, -1, 1, 3, -1]

arr: [6, 8, 0, 3, 1]

Op: [8, -1, 3, -1, -1].

BF: i: 0 → n-1

j: i+1 → n-1.

TC = O(n²)

AS = O(1)

[8, 3, 7, 9]

9 7 9 -1

max = 9

9 9 -1

l k k k k k

[6, 8, 0, 1, 3]

8 -1 1 3 -1

LIFO

3, 1, 0, 8, 6

Stack.

```
vector<int> nextGreater(vector<int> arr) {
    int n = arr.size();
    stack<int> s;
    vector<int> result(n);

    for (int i = n - 1; i >= 0; i--) {
        while (!s.empty() && arr[i] >= s.top()) {
            s.pop();
        }
        if (s.empty()) {
            result[i] = -1;
        } else {
            result[i] = s.top();
        }

        s.push(arr[i]);
    }
    return result;
}
```

→ O(n)

→ O(n)

Overall TC = O(n + n) = O(n).

l k

9 8 7 6

l k k k

6 7 8 9

7 8 9 -1

6
7
8
9

[10, 20, 30, 50, 10, 70, 30]

K=1 10, 20, 70, 50, 10, 70, 30 70

K=2 10, 20, 30, 10, 10, 30 30

K=3 10, 20, 10, 10, 10 20

K=4 10, 10, 10, 10 10

K=5 10

6 10

7 10

Op: [70, 30, 20, 10, 10, 10, 10].