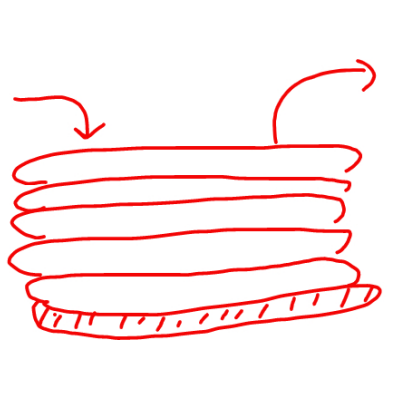
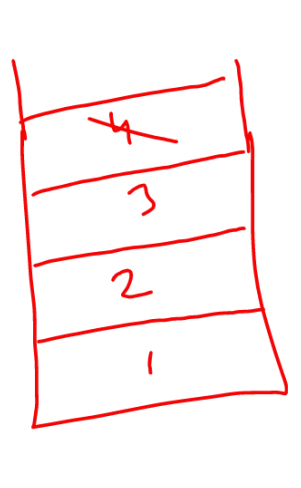
STACKS

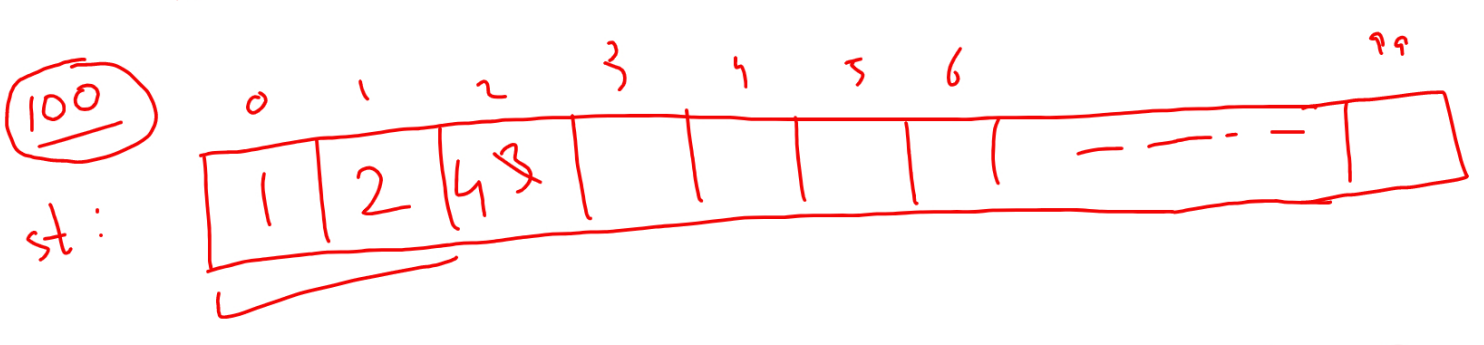


LIFO => Last In First Out



Push(4)
Pop()
top() → 3

st.top()

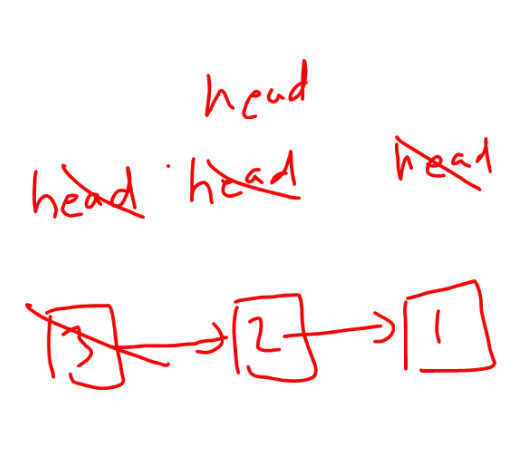


arr[size]

size = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 2

Push(1)
Push(2)
Push(3)
Pop()
Push(4)

size



Push(1)
Push(2)
Push(3)
← Pop()

(head = head→next)

{ [}] => Unbalanced

[({ })] [ ] balanced

{ [ ] } balanced → True.

LIFO

{ [ ] } (

{
~~{~~ → True.
~~(~~
~~[~~

~~(~~
~~{~~
stack

(
~~(~~ → False.
~~{~~

Time Complexity: O(len)
Aux. space: O(len)

"[{((((("

=> (( )) True

((( )) False.

) (

op: ~~1~~ 2
cl: ~~0~~ ~~1~~ 2

op: ~~1~~ ~~2~~ 3
cl: ~~0~~ ~~1~~ 2

op: 0
cl: ~~0~~ 1
=> False

op < cl → False.

[{] }   1/0

1 & (1 | 0)

(1 & (0 | 1))

(1 & (0 & (1 | 1)))
1

~~10~~
11

~~6~~
3
0

I/P: [6 8 0 1 3]
O/P: [8 -1 1 3 -1]

I/P [6 5 7]
O/P [7 7 -1]

i: 0 → n-1
j: i+1 → n-1
O(n²)

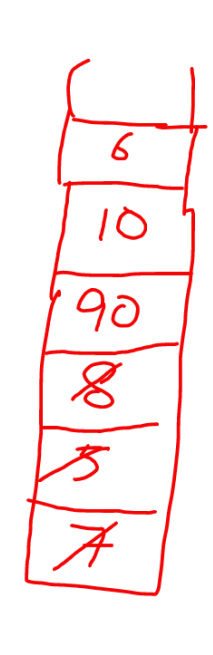
i = ~~0~~ ~~1~~ 2
[j = ~~1~~ ~~2~~ ~~2~~ 3]

[~~7 5 7~~]

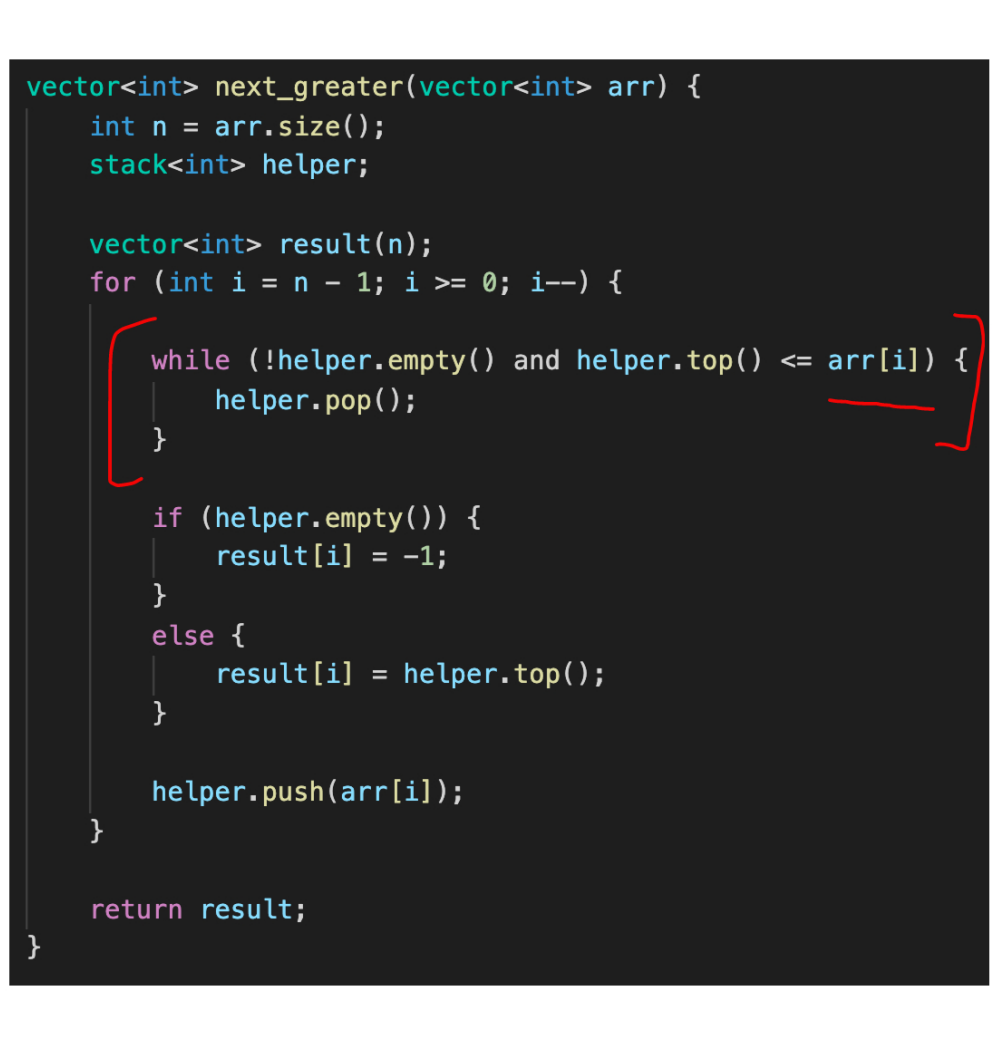
10 90 100
90 100 -1

I/P [6 10 90 8 5 7]
O/P = [10 90 -1 -1 7 -1]

O(n)

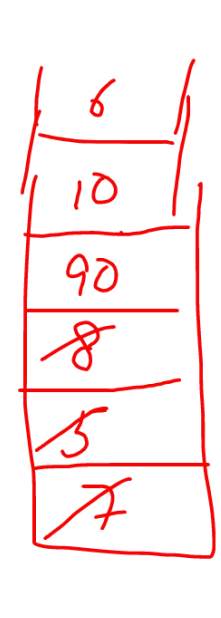


TC: O(n)
Aux. space: O(n)



```
vector<int> next_greater(vector<int> arr) {
    int n = arr.size();
    stack<int> helper;

    vector<int> result(n);
    for (int i = n - 1; i >= 0; i--) {

        while (!helper.empty() and helper.top() <= arr[i]) {
            helper.pop();
        }

        if (helper.empty()) {
            result[i] = -1;
        }
        else {
            result[i] = helper.top();
        }

        helper.push(arr[i]);
    }

    return result;
}
```

7

[6 10 90 8 5 7]
[10 90 -1 -1 7 -1]