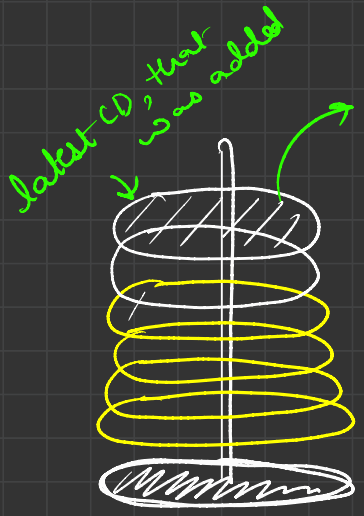
Stacks

* Linear Data Structures
  - ↳ Arrays
  - ↳ Array Lists
  - ↳ Linked List
  - ↳ Queues

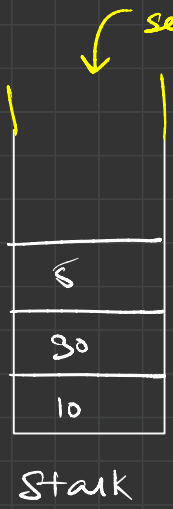
* Non Linear Data Structures
  - ↳ HashMap | HashSet
  - ↳ tree - tries - heaps
  - ↳ graph

Stacks



LIFO

{ last in first out }



Stack (LIFO)

- push(int v): Add a value v to the top of the stack
- pop(): Remove the topmost Ele
- peek(): able to view the topmost data.
- size(): size of the stack

```
class Stack
{
    ArrayList<Integer> list;
    int size;
    Stack()
    {
        list = new AL<>();
        size = 0;
    }

void push (int v)   TC: O(1)
    {
        list.add(v);
        size++;
    }
```

```
int pop()   TC: O(1)
    {
        if(size > 0)
        {
            int ele = list.remove(size-1);
            size--;
            return ele;
        }
        else
            return -1;
```

```
int peek()   TC: O(1)
    {
        if(size > 0)
        {
            return list.get(size-1);
        }
        else
            return -1;
    }
```

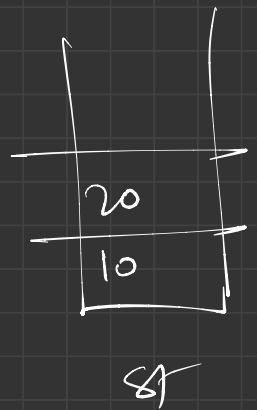
Practical Examples of Stack

- Recursion
- Memory management
- Cache
- Android back

```
Stack<Integer> st = new Stack<>();
```

class

reference variable




```
st.push(10);
st.push(20);
st.push(30);
print(st.peek()); ---> 30
st.pop(); ---> 30
print(st.peek()); ---> 20
st.size(); ---> 2
```

Q. Extra Brackets.

string str = "(a+b)"; → No

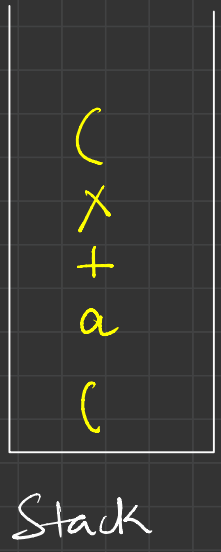
"((a+b))"; → yes

"(a+b)+(c+d+(e*f)())" → yes

"((a)+(b))"; → No

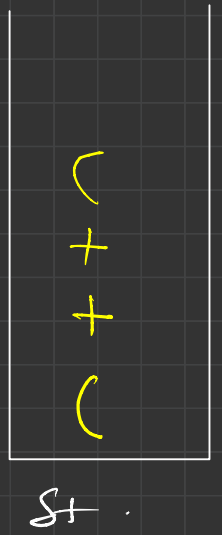
Str = "(a+(b*d+f-(m)+n-o)x(z)())"

last opend is first closed!



```java
public boolean ExtraBrackets(String exp) {
  // Write your code here
  Stack<Character> st = new Stack<>();

  for (int i = 0; i < exp.length(); i++) {
    char ch = exp.charAt(i);

    if (ch != ')') {
      st.push(ch);
    } else {
      if (st.peek() == '(') {
        return true;
      } else {
        // remove the exp in between
        while (st.peek() != '(') {
          st.pop();
        }

        // as this pair holds a exp in between, that means a valid bracket
        // so remove the corresponding opening bracket
        st.pop();
      }
    }
  }

  // if no extra bracket was found
  return false;
}
```

exp = "((a)+(b)+((c+d)))"



TC: O(N)
SC: O(N)

N → push()
N → pop()

O(N) + O(N) ≈ O(N)

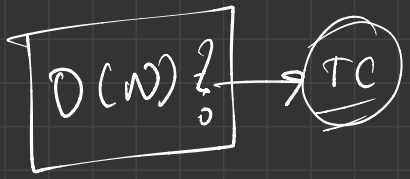
Next Greater Element On Right

	0	1	2	3	4	5	6	7	8	9
arr[] = {	3	6	1	2	7	3	4	1	2	5 }
nger[] = {	6	7	2	7	-1	4	5	2	5	-1 }

Brute force

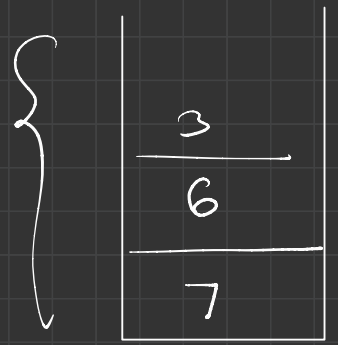
TC : $O(N^2)$

SC : $O(1)$



arr[] = { 3, 6, 1, 2, 7, 3, 4, 1, 2, 5 }

nger[] = { 6, 7, 2, 7, -1, 4, 5, 2, 5, -1 }



Stack → all the potential nge

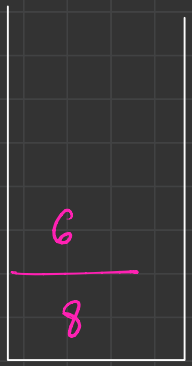
```java
public static long[] nextLargerElement(long[] arr, int n) {
  //Write code here and print output
  Stack<Long> st = new Stack<>();
  long[] nger = new long[n];

  // traverse from right to left
  // because I want info about potential nge
  for (int i = n - 1; i >= 0; i--) {
    // remove smaller potential nge from stack
    // as I'm covering for them
    while (st.size() > 0 && st.peek() <= arr[i]) {
      st.pop();
    }

    if (st.size() == 0) {
      nger[i] = -1;
    } else {
      nger[i] = st.peek();
    }

    st.push(arr[i]);
  }

  return nger;
}
```

is life time

O(N)

arr[] = { 6, 8, 0, 1, 3 }

{ 8, -1, 1, 3, -1 }

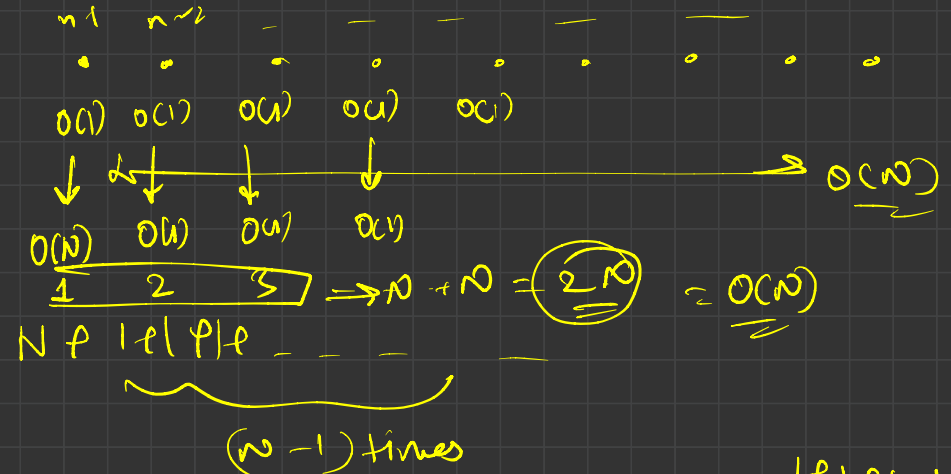


St

Amortized O(N)

TC : O(N)

SC : O(N)

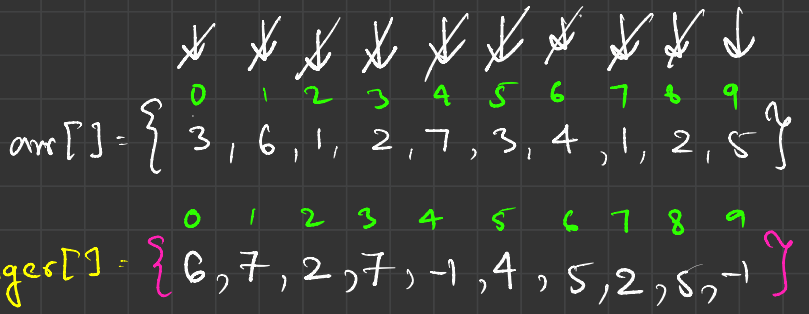

n1 n2
O(1) O(1) O(1) O(1) O(1)
O(N)
O(N) O(1) O(1) O(1)
1 2 3
→N + N = 2N = O(N)
N + 1 + 1 + 1 + ...
(N-1) times
1+1+1+...
N + (N-1) = (2N-1)
O(N)

arr[] = { 3, 6, 1, 2, 7, 3, 4, 1, 2, 5 }

nger[] = { 6, 7, 2, 7, -1, 4, 5, 2, 5, -1 }

Stack → pool of people looking for nger

```java
public static long[] nextLargerElement(long[] arr, int n) {
  //Write code here and print output
  Stack<Integer> st = new Stack<>();
  long[] nger = new long[n];

  // traverse from left to right
  // because I want to create a pool of people looking for nge
  for (int i = 0; i < n; i++) {
    long ele = arr[i];

    // I can be a nge
    // go and see in pool, who all intrested
    while (st.size() > 0 && arr[st.peek()] < ele) {
      nger[st.peek()] = ele;
      st.pop();
    }

    //   I'm also looking for my nge
    st.push(i);
  }

  // people still looking for nge, will have it as -1
  while (st.size() > 0) {
    nger[st.peek()] = -1;
    st.pop();
  }

  return nger;
}
```



TC : O(N)
SC : O(N)

Stock Span Problem

0	1	2	3	4	5	6
100	80	60	70	60	75	85

ngeli

-1, 0, 1, 1, 3, 1, 0

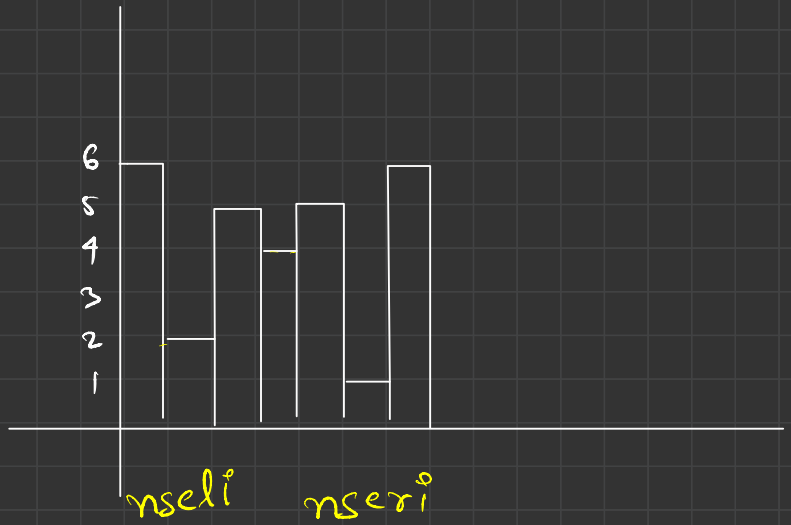
Stopping index

dist

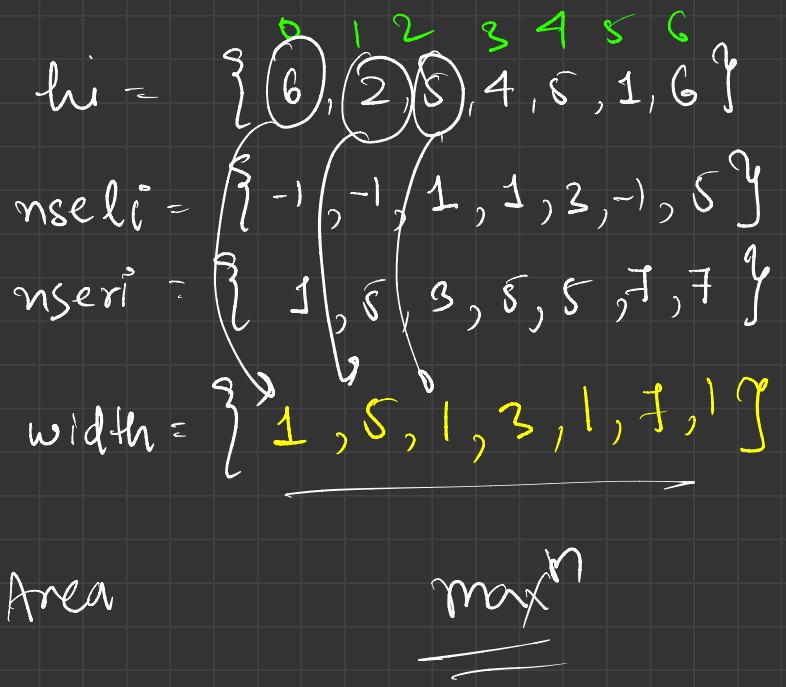
0-(-1)	1-0	2-1	3-1	4-3	5-1	6-0
=1	=1	=1	=2	=1	=4	=6

→ Ans

Largest Area Histogram



maxArea = ~~6~~ ~~10~~ 12


0 1 2 3 4 5 6
hi = {6, 2, 5, 4, 5, 1, 6}
nseli = {-1, -1, 1, 1, 3, -1, 5}
nseri = {1, 5, 3, 5, 5, 7, 7}
width = {1, 5, 1, 3, 1, 7, 1}

Area

max

① get NSEL I

② get NSE R I

③ get width right - left - 1

Area = width × height

④ store Max Area