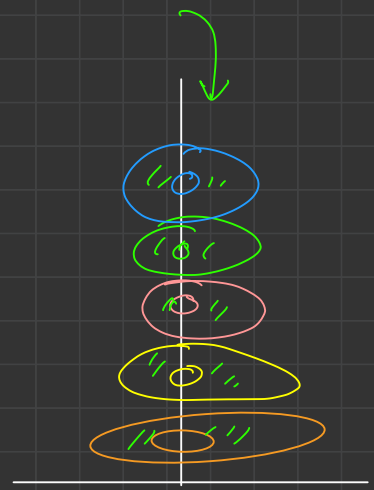
Stacks.

☆ linear data structures

1. Strings
2. Arrays
3. Array List
4. Linked List
5. doubly linked list
6. String Builder

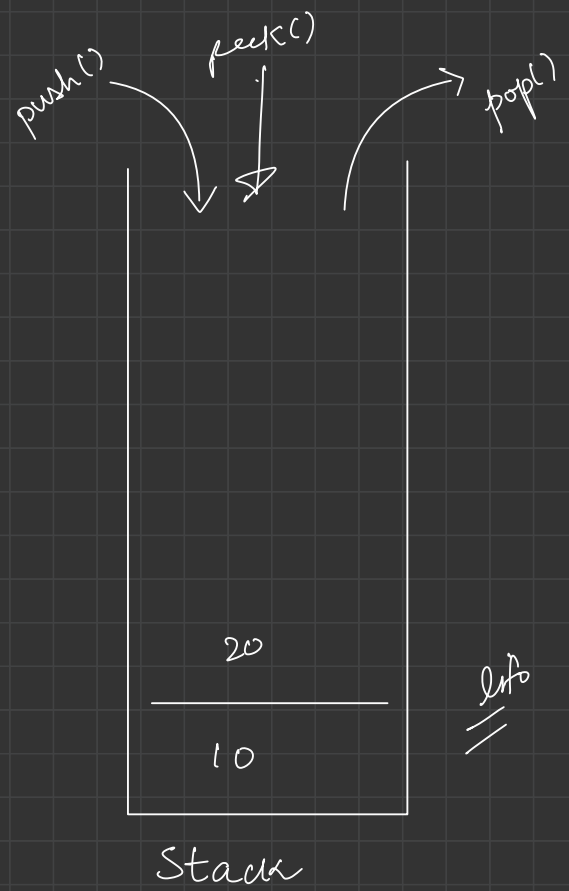
Stacks (last in, first out) (lifo)

↳ linear data structure



Stack of CD

- add to the topmost pos.
- remove from the topmost pos




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

Add Element to the topmost pos of stack

push()

remove Element from the topmost pos of stack

pop()

```
st.push(10)
st.push(20)
st.push(30)
st.pop()
```

LIFO

allows to view topmost ele.

peek()

gives size of stack

size();

Practical Use Case of Stacks

- → Recursion ((callstack)
- → Memory Managment
- → Redo/undo functionality
- → Cache

Q Extra brackets

String str : "(a + b)" → No

: "((a+b))" → yes, we have extra brackets here

: (a+b) + ((+d * e * (f / g) ()) → yes

: ((a) + (b)) → No

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

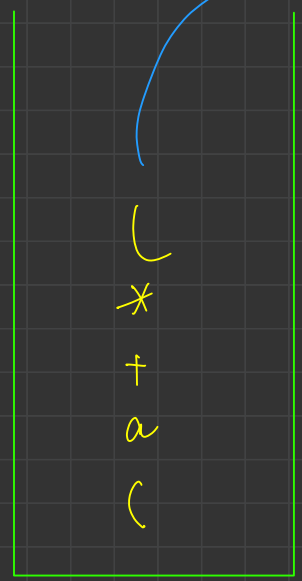
1 ~~2~~ ~~3~~

{ last bracket opened, is the first bracket to be closed. (lifo) }

Stack!

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

Extra Pair!



Stack

① add everything to the stack, until found a closing bracket.

② try finding corresponding open. bracket in stack.

③ if have a exp in b/w the rest an extra pair

Next Greater Element On Right

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: $O(1)$

TC: O(N) ?

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

{ 6 7 2 7 -1 4 5 2 5 -1 }

nger

stack

giving potential nger

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

5 2 5 -1

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

long[] nger = new long[n];

for (int i = n - 1; i >= 0; i--) {
    // remove smaller people in stack, as they can't be
    while (st.size() > 0 && st.peek() <= arr[i]) {
        st.pop();
    }

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

    st.push(arr[i]);
}

return nger;
```

```java
Stack<Long> st = new Stack<>();

long[] nger = new long[n];

for (int i = n - 1; i >= 0; i--) {
    // remove smaller people in stack, as they can't be
    while (st.size() > 0 && st.peek() <= arr[i]) {
        st.pop();
    }

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

    st.push(arr[i]);
}

return nger;
```

O(N) life time

N pop at max

N push

$$\sum \boxed{a + b + c \ldots\ldots\ldots\ldots \alpha}$$

N operation of for loop

No. of time while loop ran in this instance

$= N$

$$\boxed{\begin{cases} TC : O(N) \\ SC : O(N) \end{cases}}$$

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