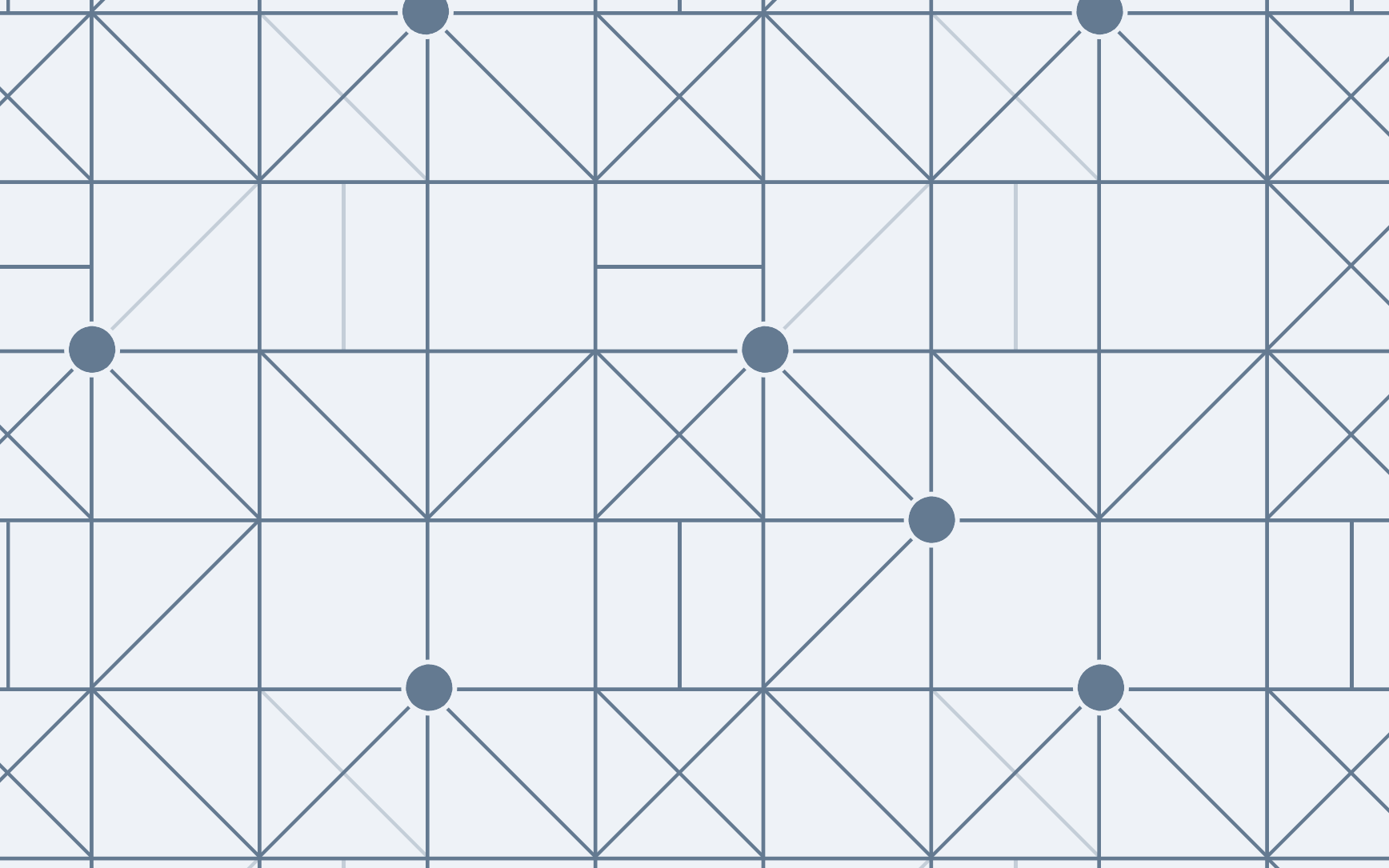
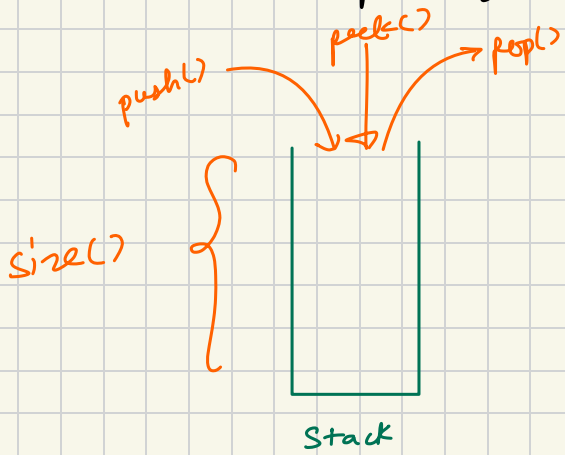


Stacks

- linear data structure
- LIFO {Last in, first out}



Agenda:

1. Extra Bracket
2. Next greater element on Right
3. Stock Span
4. Largest histogram area

Extra Brackets

T.C : O(N), S.C : O(N)

NOTE : Bracket pairs will be always balanced and in correct order

string str = "(a+b)" — NO Extra Bracket

= "((a+b))" — YES Extra Bracket

NOTE:
A bracket pair is useful if it have a new expression inside himself.

= "(a+b) * (c+d+(e+f) / ())" — YES Extra Bracket

= "((a)+(b))" — NO Extra Bracket

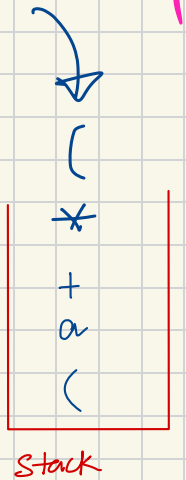
str = "(a + (b * d + f - (m) + 0) * (p)( ))"

Closing Bracket

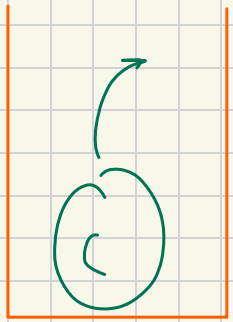
try to find corresponding opening bracket,

& they have an exp. inb/w hence useful pair

No exp. removed, hence extra bracket pair

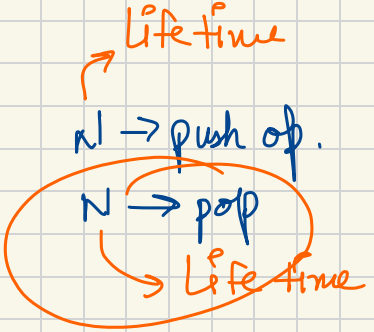


str = "(((a+b)+(c)))"



Extra bracket pair!

```
for (int i=0 ⟶ n)
{
    work done;
}
```



$i=0 \quad i=1 \quad i=3 \quad \cdots\cdots \quad i=n-1$

$\sum_{1}$ $\alpha \quad \beta \quad \gamma \quad \cdots \quad \phi$

$\sum_{1}$ $a \quad b \quad c \quad \cdots \quad = N!$

TC : O(N)

Next greater element on right

person nearest to me in right-hand array greater than me

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

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

Brute force

```
for ( int i=0 → n)
    { for(int j=i+1 → n)
```

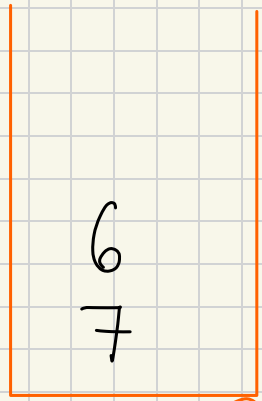
TC : O(N^2)
SC : O(1)

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

0 1 2 3 4 5 6 7 8 9

{ 6 7 2 7 -1 4 5 2 5 -1 } → nger

TC: O(N)

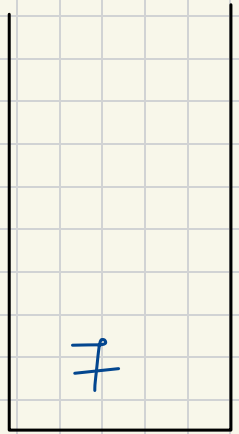


Stack { potential nger }

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

0 1 2 3 4 5 6 7 8 9

-1 4 5 2 5 -1



stack { potential nger }

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

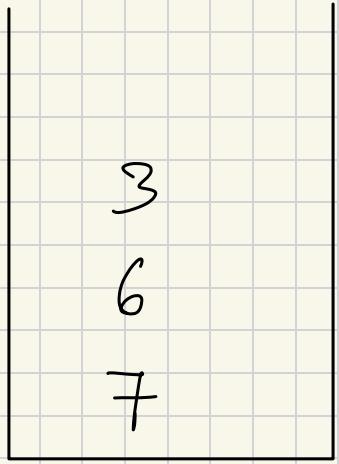
0 1 2 3 4 5

6 7 2 7 -1 -1

TC : O(N)

SC : O(N)

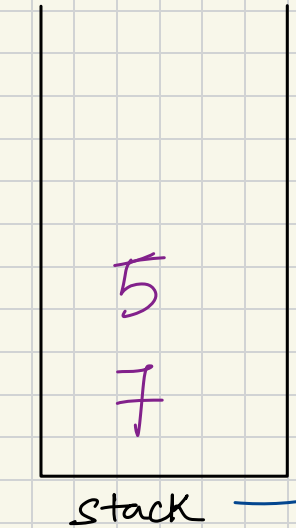
```java
public static long[] nextLargerElement(long[] arr, int n) {
    // potential nger
    Stack<Long> st = new Stack<>();

    long[] nger = new long[n];

    // right to left
    for (int i = n - 1; i >= 0; i--) {
        long ele = arr[i];

        while (st.size() > 0 && ele >= st.peek()) {
            // these are useless potential ans in the stack
            st.pop();
        }

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

        st.push(ele);
    }

    return nger;
}
```



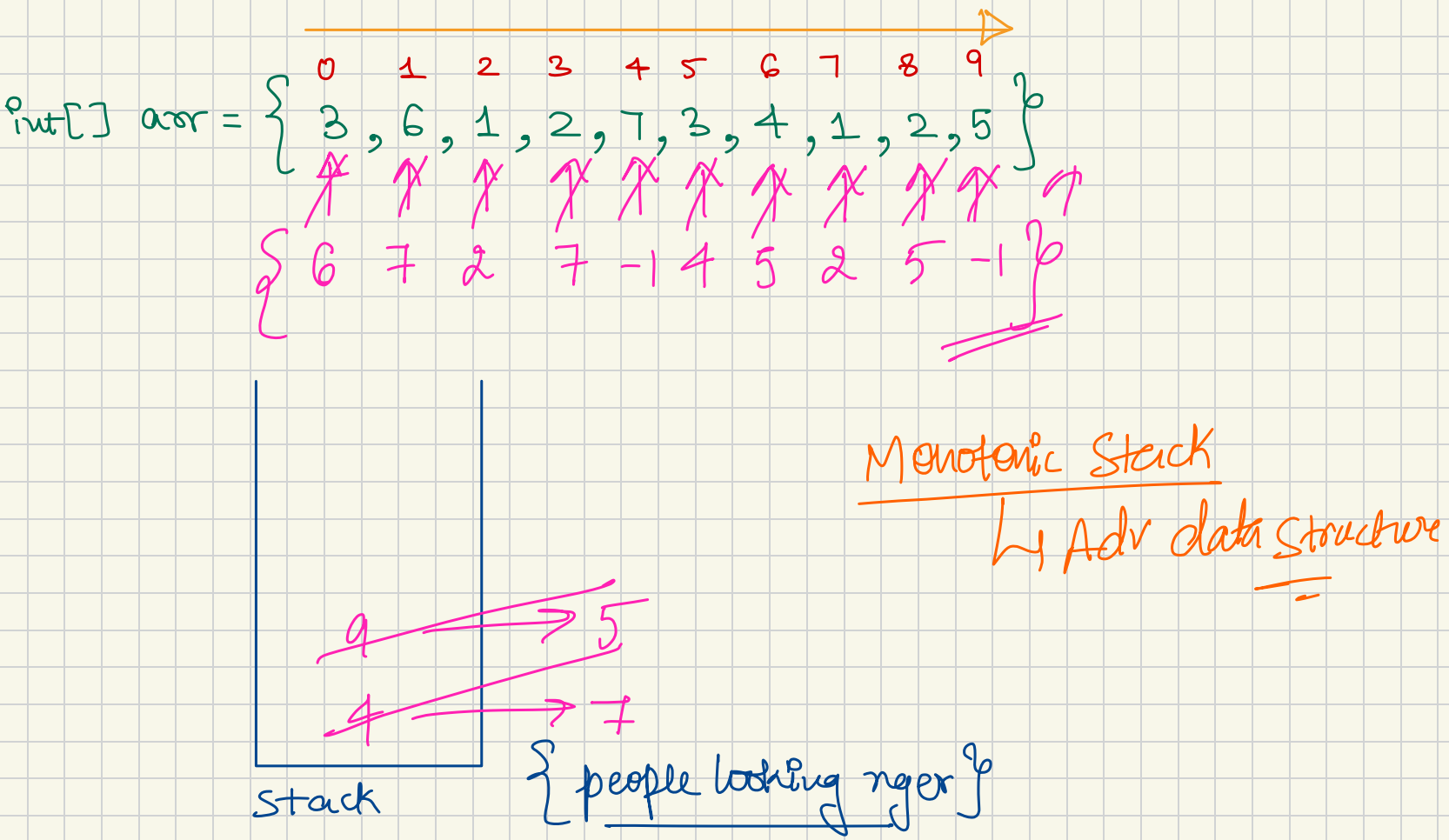
Approach 2.

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

0	1	2	3	4	5	6	7	8	9
3	6	1	2	7	3	4	1	2	5
6	7	2	7	-1	4	5	2	5	-1

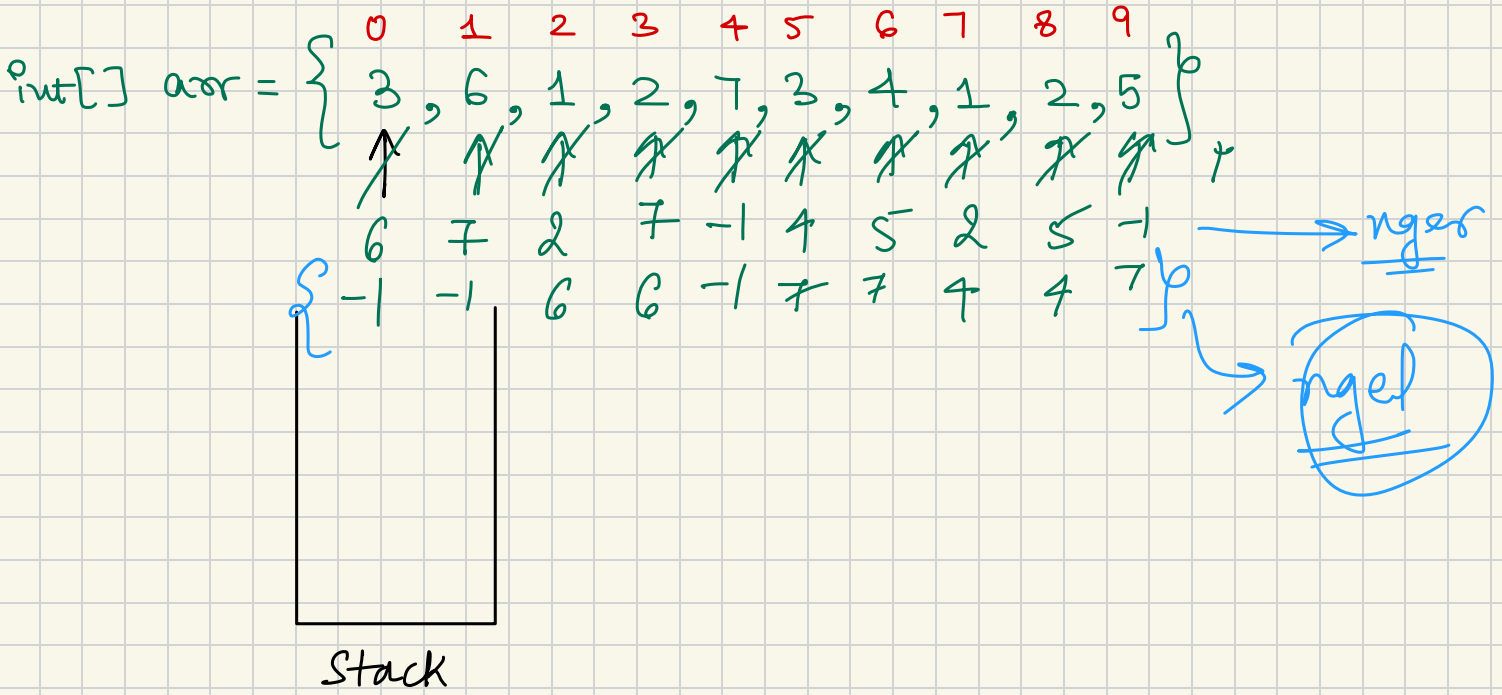


{ people looking for nges }



Monotonic Stack

↳ Adv data structure


int[] arr = { 3, 6, 1, 2, 7, 3, 4, 1, 2, 5 };
0 1 2 3 4 5 6 7 8 9
6 7 2 7 -1 4 5 2 5 -1
-1 -1 6 6 -1 7 7 4 4 7
nger
nge
Stack

Stock Span Problem

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

arr[] = {100, 80, 60, 70, 60, 75, 85}

span ⟶ no. of consecutive day, Inc. current day stock price less than equal to current day price

$$arr[] = \{\overset{0}{100}, \overset{1}{80}, \overset{2}{60}, \overset{3}{70}, \overset{4}{60}, \overset{5}{75}, \overset{6}{85}\}$$

$$span[] = \{1, 1, 1, 2, 1, 4, 6\}$$

Brute force

TC: $O(N^2)$ SC: $O(1)$

arr[] = { 100, 80, 60, 70, 60, 75, 85 }

0 1 2 3 4 5 6

nge[] = { -1, 0, 1, 1, 3, 1, 0 }

span[] = { 1, 1, 1, 2, 1, 4, 6 }

dist = i - nge