JAVA & DSA Series

SKILLS

Stacks

Part - 2

Raghav Garg

COLLEGE WALLAH

Today’s checklist

- Interview questions related to stacks

Q. Balanced brackets

Check whether a given bracket sequence is balanced or not

String str = "()(())" True

str = "(()()(" False

str = "(()())" True

str = ")()())" False

str = "(())()" False

str = "()(())" True

Rules

1) Opening → push

2) Closing

a) st top → (→ pop

End me :

If the st is empty,

they are balanced → true

str = " (( ) ( ) ("

i

Rules

1) Opening → push

2) Closing

a) st top → (→ pop

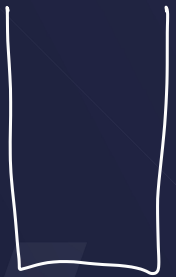
st

if (st.size() != 0)

return false

str = "(())()"

st



Rules

1) Opening → push

2) Closing

a) st top → (→ pop

b) if st is empty -
return false

Practice → Follow Homework

Find the minimum number of brackets that we need to remove to make the given bracket sequence balanced.

str = "(()()(" False

str = ")()())" False

str = "(())())" False

Practice - LeetCode (LeetCode - 20)

Check whether a given bracket sequence is valid or not

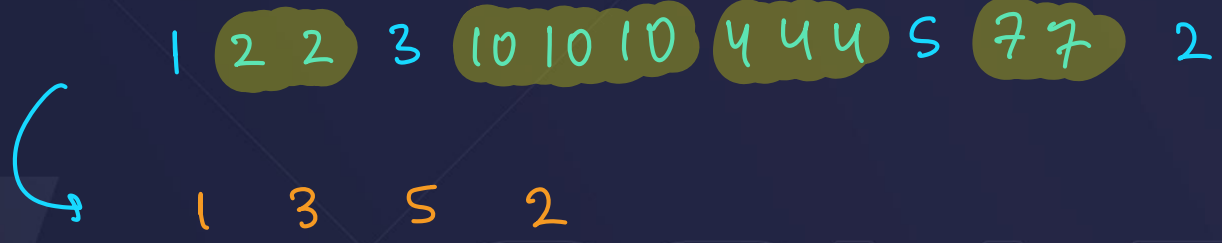
([ ]) { } Valid

([ )] Invalid

push → '(' || '{' || '['

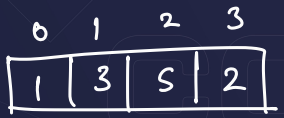
Q. Remove consecutive subsequences

Given a sequence of numbers. Remove all the consecutive subsequences of length greater than or equal to 2 that contains the same element.



SKILLS
COLLEGE WALLAH

Arr 1 2 2 3 10 10 10 4 4 4 5 7 7 2




```
int[] res = new int[st.size()];
```

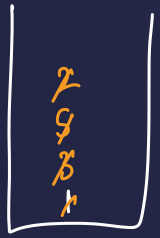
```
if(st.size()==0) push
if(st.peek() != arr[i]) push
if(st.peek() == arr[i]){
    if(arr[i] != arr[i+1])
        st.pop();
    if(arr[i] == arr[i+1]){
        do nothing
    }
```

```
}
public static void main(String[] args) {
    int[] arr = {1,2,2,3,10,10,10,4,4,4,5,7,7,2};
    int[] res = remove(arr);
    for(int i=0;i<res.length;i++){
        System.out.print(res[i]+" ");
    }
}
public static int[] remove(int[] arr){
    int n = arr.length;
    Stack<Integer> st = new Stack<>();
    for(int i=0;i<n;i++){
        if(st.size()==0 || st.peek()!=arr[i])
            st.push(arr[i]);
        else if(arr[i]==st.peek()){
            if(i==n-1 || arr[i]!=arr[i+1]) st.pop();
        }
    }
    int[] res = new int[st.size()];
    int m = res.length;
    for(int i=m-1;i>=0;i--){
        arr[i] = st.pop();
    }
    return res;
}
```

arr

1 2 2 3 10 10 10 4 4 4 5 7 7 2

i n-1



res

m = 4

Q. Next greater element → (+ve)

input	1	3	2	1	8	6	3	4
res	3	8	8	8	-1	-1	4	-1

Without Extra Space → MI → T.C. → $O(n^2)$ S.C → $O(1)$

$O(n)$ → Solve → using stacks

Q. Next greater element

	i							
input	1	5	3	2	1	6	3	4
res	5	6	6	6	6	-1	4	-1

```
for(int i = n-2; i >= 0; i--){
    while( st.peek() < arr[i] && st.size()>0)
        st.pop();
    if(st.size()==0) res[i] = -1;
    else res[i] = st.peek();
    st.push(arr[i]);
}
```

<=n

Q. Next greater element

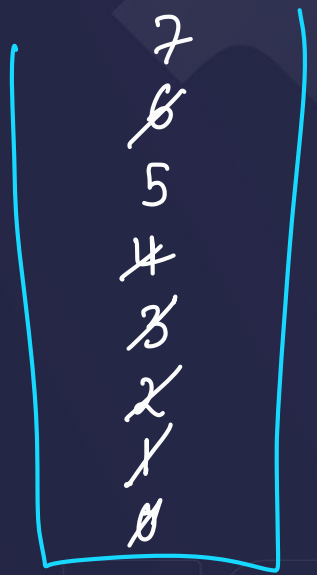
input 1 5 3 2 1 6 3 4

pop , ans mark , push

Q. Next greater element

	0	1	2	3	4	5	6	7
input	1	5	3	2	1	6	3	4

res	5	6	6	6	6	-1	4	-1



2nd approach

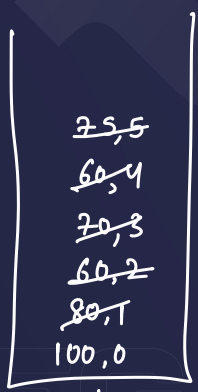
Practice

GFG → Stock span problem

Given a series of N daily price quotes for a stock, we need to calculate the span of the stock's price for all N days. The span of the stock's price in one day is the maximum number of consecutive days (starting from that day and going backward) for which the stock price was less than or equal to the price of that day.

SKILLS
COLLEGE WALLAH

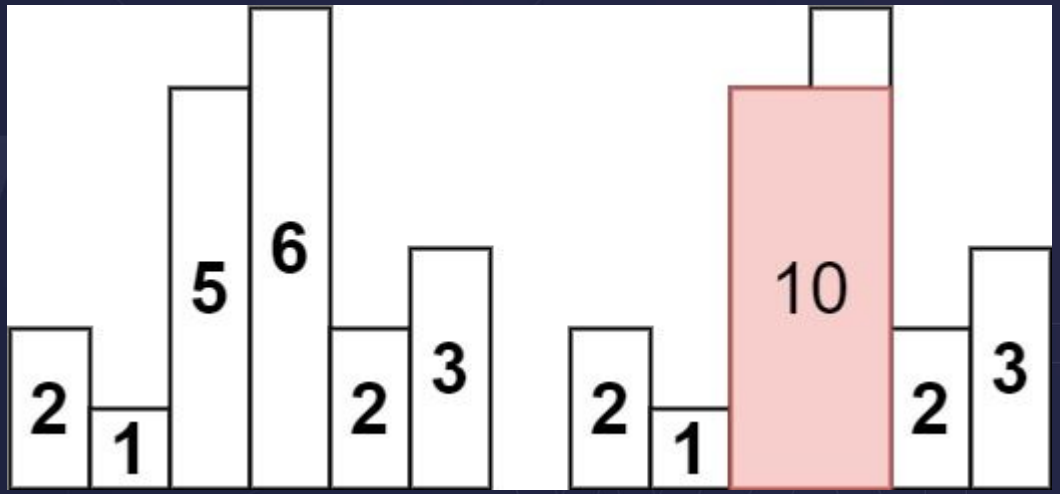
0	1	2	3	4	5	6
100	80	60	70	60	75	85
1	1	1	2	1	4	6

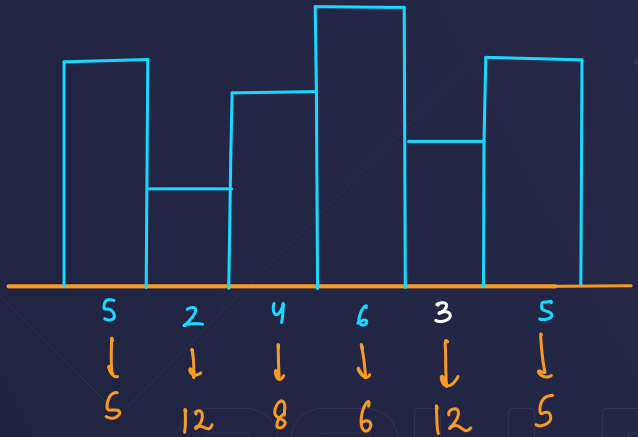


Q. Largest rectangle in Histogram

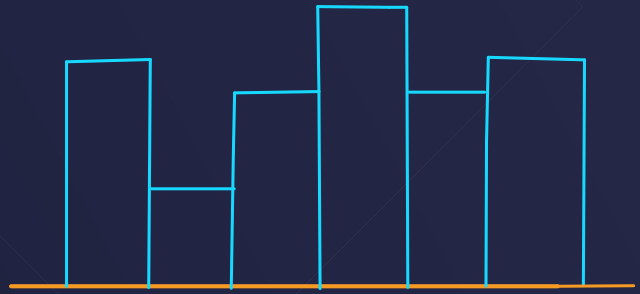
O(n)

Given an array of integer heights representing the histogram's bar height where the width of each bar is 1, return the area of the largest rectangle in the histogram.





PSE & NSE



	5	2	4	6	3	5
	0	1	2	3	4	5
nse[]	1	6	4	4	6	6
pse[]	-1	-1	1	2	1	4

st

(5, 5)
(3, 4)
~~(6, 3)~~
~~(4, 2)~~
(2, 1)
~~(5, 0)~~

st

(2, 1)
~~(4, 2)~~
~~(6, 3)~~
~~(3, 4)~~
~~(5, 5)~~

$$(heights[4]) * (nse[4] - pse[4] - 1)$$

Next Smaller Element

h 2, 1, 5, 6, 2, 3

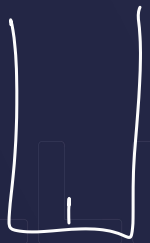
nse 6 6

```
int n = heights.length;
Stack<Integer> st = new Stack<>();
int[] nse = new int[n];
int[] pse = new int[n];
// calculate nse[]
st.push(n-1); // index
nse[n-1] = n;
for(int i=n-2;i>=0;i--){
    while(st.size()>0 && heights[st.peek()]>heights[i]){
        st.pop();
    }
    if(st.size()==0) nse[i] = n;
    else nse[i] = st.peek();
}
```

```
int n = heights.length;
Stack<Integer> st = new Stack<>();
int[] nse = new int[n];
int[] pse = new int[n];
// calculate nse[]
st.push(n-1); // index
nse[n-1] = n;
for(int i=n-2;i>=0;i--){
    while(st.size()>0 && heights[st.peek()]>heights[i]){
        st.pop();
    }
    if(st.size()==0) nse[i] = n;
    else nse[i] = st.peek();
}
```

heights 1, 1
0 1

nse 2



Q. Min Stack [special stack implementation]

st.peek() → O(1)

st.push() → O(1)

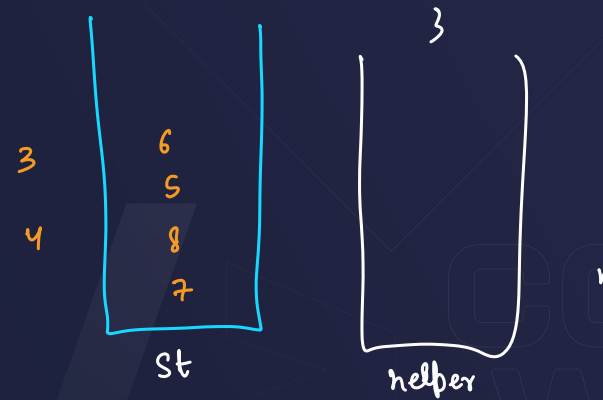
st.pop() → O(1)

st.getMin() → O(1)

SKILLS
COLLEGE WALLAH

SKILLS

```
int getmin(){
    Stack helper;
    int min = max;
```



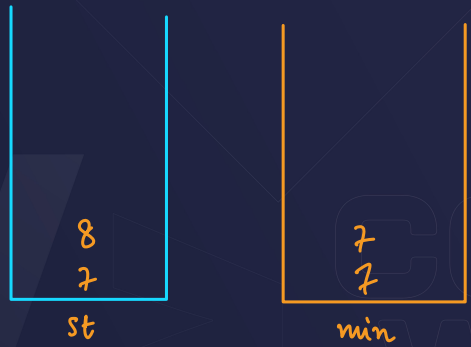
COLLEGE WALLAH

Approach - 2 : getmin() → O(1)

S.C. → O(n)

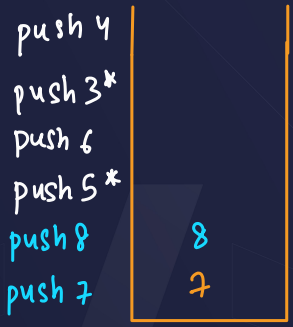
push, pop, peek, getmin → O(1)

7 8 5 6 3 4



Approach - 3 : without extra stack

variable → one



Val

min = ~~7~~ ~~8~~ ~~3~~ ~~5~~ 7

min - oldmin = st.peek()

oldmin = min - st.peek()

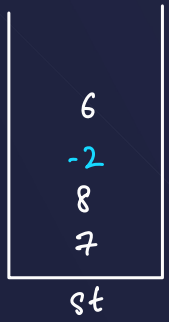
min = oldmin

```
if(val < min)
   st.push(val - min)

getmin()
   return min;

pop(){
   if(st.peek() > min)
         st.pop()
   if(st.peek() < min)
      int old = min - st.peek;
      min = old;
      st.pop()
```

```
void push(int val){
    if(st.size()==0){
        st.push(val);
        min = val;
    }
    if(val > min)
        st.push(val);
    if(val < min){
        st.push(val-min);
        min = val;
    }
}
```

min = ~~7~~ 3

2* val - min



min = ~~7~~ ~~5~~ ~~3~~ 5

```
void pop(){
    if(st.size==0) return;
    if(st.peek() > min)
        st.pop();
    if(st.peek() < min){   //restore
        int old = min - st.peek();
        min = old;
        st.pop();
    }
}
```

old = 2*min - st.peek();

3

SKILLS

```
["MinStack","push","push","push","push","getMin","pop","getMin","pop","getMin","pop","getMin"]

[[],[2],[0],[3],[0],[],[],[],[],[],[],[]]
```

min = 1 2

Output

```
[null,null,null,null,null,0,null,0,null,2,null,2]
```

Expected

```
[null,null,null,null,null,0,null,0,null,0,null,2]
```

COLLEGE WALLAH

min = ~~7~~ ~~5~~ ~~3~~ 5

```
int getMin(){
    if(st.size()==0) return -1;
    return min;
}
```

```
int top(){
    if(st.size()==0) return -1;
    if(st.peek() > min)
        return st.peek();
    if(st.peek() < min){ //farzi
        return min;
    }
}
```

Input

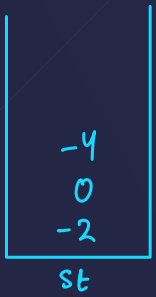
```
["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[]]
```

Output

```
[null,null,null,null,-3,null,0,-3]
```

Expected

```
[null,null,null,null,-3,null,0,-2]
```



val
-3

min = 1 -2 -3

2 * val - min

2 * (-3) - (-2)

-6 + 2 = -4

-2 0 -3

min = ~~-2~~ -3

2*(-3) - (-2) = -4

st.peek() = 2*newMin - oldmin

Oldmin = 2*min - st.peek()

SKILLS

val < min ⟶ val - min < 0

-3 -2

val - min ≮ val

2* val - min < val

1

val + (val - min)

val + (-ve) < val

COLLEGE WALLAH