


Stack
LIFO → last in, first out
semantics

⇒ Syntax

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

⇒ function

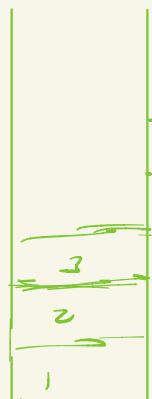
1) add) st.push(5)

2) remove) st.pop() → remove and return the top element

3) int a = st.peek(); → returns the top element

4) size) st.size();

```
public class Example {  
    Run | Debug  
    public static void main(String[] args) {  
        Stack<Integer> st=new Stack<>();  
  
        st.push(item: 1);  
        st.push(item: 2);  
        st.push(item: 3);  
  
        System.out.println(st.peek());  
  
        st.push(item: 4);  
        st.push(item: 5);  
  
        while(st.size()>0){  
            int a=st.pop();  
            System.out.println(a);  
        }  
    }  
}
```



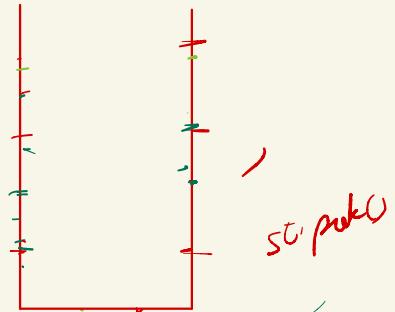
int a = st.pop()

5
4
3
2
1

~~→ { [(]) } X → balanced~~

~~→ { { { [] } } () } ✓~~

int a=5;?
if (a==s) return false



```
public boolean isValid(String s) {
    Stack<Character> st=new Stack<>();
    for(int i=0; i<s.length(); i++){
        char ch=s.charAt(i);
        if(ch=='(' || ch=='{' || ch=='['){
            st.push(ch);
        } else if(ch==')'){
            if(st.size()==0 || st.peek()!='(') return false;
            st.pop();
        } else if(ch=='}'){
            if(st.size()==0 || st.peek()!='{') return false;
            st.pop();
        } else if(ch==']'){
            if(st.size()==0 || st.peek()!='[') return false;
            st.pop();
        }
    }
    if(st.size()==0) return true;
    return false;
}
```

Ques input string ()
a=x+x+x+x+0
⇒ ↓

() ()
() () ()
a=x+x+x+x+x+0
() ()

O(1) space
O(N) time

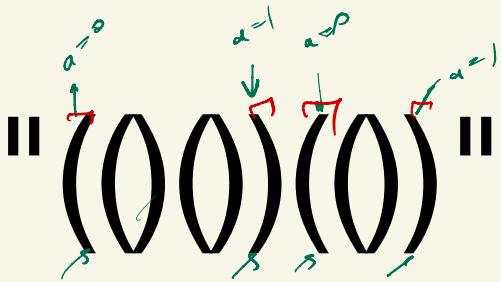
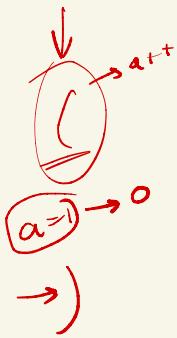
→ if (ch=='(') a++;
else a--;

if (a<0)
return false

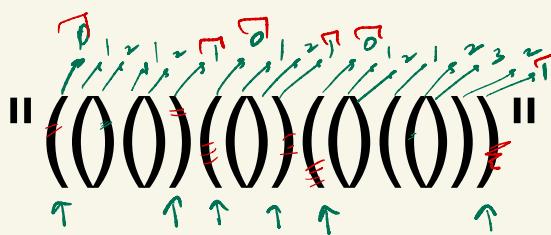
() ()
a=x+x+x+0

if (a==0)
return true

a=1



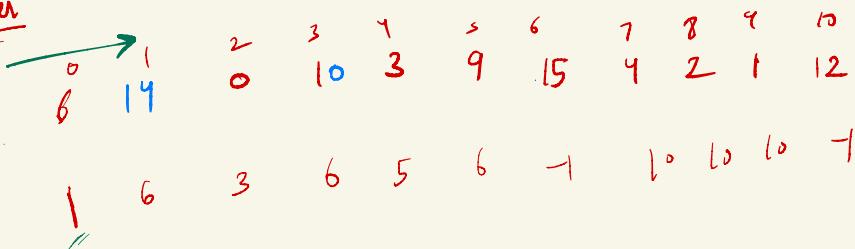
$a \hookrightarrow$ number of opening brackets with no corresponding closing bracket



if ($ch == '('$
and $a == 0$)

if ($ch == ')'$
and $a == -1$)

Next greater

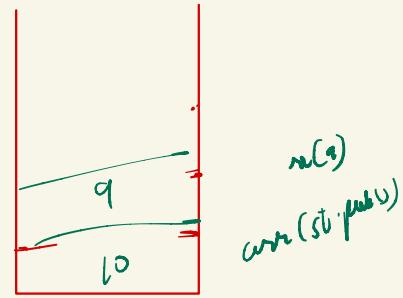


```
// Next greater on right
public static long[] nextLargerElement(long[] arr, int n){
    long[] ngr=new long[n];
    Stack<Long> st=new Stack<>();

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

        while(st.size()>0 && st.peek()<=ele){
            st.pop();
        }

        if(st.size()==0){
            ngr[i]=-1;
        } else {
            ngr[i]=st.peek();
        }
        st.push(ele);
    }
    return ngr;
}
```



$\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 6, 5, 3, 4, 3 \end{matrix}$ $(2n-1 \rightarrow 0)$

$\begin{matrix} -1 & 6 & 4 & 6 & 6 \end{matrix}$

$\begin{matrix} 1 & 3 & 1 & 5 & 2 & 3 \\ -1 & 5 & 5 & 7 & 3 & 7 \end{matrix}$

$\begin{matrix} -2 & 1 & 5 & 4 & 3 \\ 5 & 5 & 1 & 1 & 1 \end{matrix}$

$9 \rightarrow 0$
 $9 \cdot 5 = 45$
 $8 \cdot 5 = 35$
 $7 \cdot 5 = 25$
 $6 \cdot 5 = 15$
 $5 \cdot 5 = 0$
 $4 \cdot 5 = 10$
 $3 \cdot 5 = 15$
 $2 \cdot 5 = 10$
 $1 \cdot 5 = 5$
 $0 \cdot 5 = 0$

$0 \rightarrow \text{amara}$
 $\rightarrow \text{akbar}$
 $\rightarrow \text{anthony}$

$\begin{matrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 \end{matrix}$

	0	1	2	3
0	0	0	1	1
1	1	0	1	1
2	0	0	0	0
3	1	0	1	0

$$M(i)(j) = 0$$

\downarrow doesn't know jth person

$\textcircled{1}$ $\xrightarrow{\text{now}}$ every where 0
 $\text{a col} \rightarrow$ every where 1

```

public boolean isCelebrity(int[][] M, int a){
    int n=M.length;
    int i=a;
    for(int j=0; j<n; j++){
        if(a==j) continue;
        if(M[i][j]!=0) return false;
    }

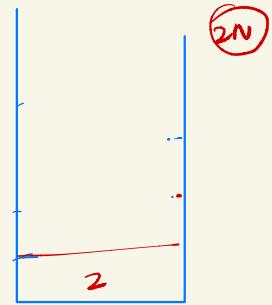
    // column check
    int j=a;
    for(i=0; i<n; i++){
        if(a==i) continue;

        if(M[i][j]!=1) return false;
    }
    return true;
}
    
```

$a = 0$
 $a = 2$

Elimination method

$\left\{ \begin{array}{l} M(a)(b) = 0 \\ \rightarrow b \text{ can't be a celebrity} \end{array} \right.$
 $\left\{ \begin{array}{l} M(a)(b) = 1 \\ a \text{ can't be celebrity} \end{array} \right.$



Sliding window maximum

O(n)

nums = [1, 3, -1, -3, 5, 3, 6, 7]

ngre \Rightarrow 1, 4, 4, 4, 6, 6, 7, 8

$k=3$

$idx = 0+2$

$n-k$

0, 1, 2, ..., $n-k$

$n-k+1$

0	1	2	3	4	5
2	3	-1	1		

$sp = 0+2$
 $ep = 2+4$

② $sp = 0 >$

$sp \leq n-k$

nums =

0	1	2	3	4	5	6	7	8	9	10
1	2	3	-1	2	1	4	1	6	1	8

 $n=11$

ngr =

0	2	3	4	5	6	7	8	9	10	
1	2	6	5	5	6	8	8	9	10	11

 $k=3$

ans =

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

```
int idx=0; // stores the maximum element index of current window
for(int sp=0; sp<=(n-k); sp++){
    if(idx<sp){
        idx=sp;
    }

    int ep=sp+k-1;

    while(ngr[idx]<=ep){
        idx=ngr[idx];
    }

    ans[sp]=nums[idx];
}

return ans;
```

```
int celebrity(int M[][], int n){
    Stack<Integer> st=new Stack<>();
    for(int i=0; i<n; i++){
        st.push(i);
    }

    while(st.size()>1){
        int a=st.pop();
        int b=st.pop();

        if(M[a][b]==0){
            st.push(a); // a can be celebrity
        } else {
            st.push(b);
        }
    }

    int a=st.pop();
    if(isCelebrity(M, a)) return a;
}

return -1;
```

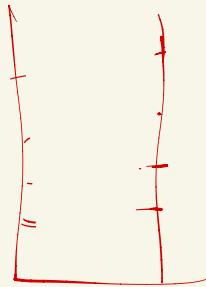
$$idx = \cancel{0} \cancel{1} \cancel{2} \cancel{3} \cancel{4} \cancel{5} \cancel{6} \cancel{7} \cancel{8} \cancel{9} \cancel{10}$$

$$sp = \cancel{0} \cancel{1} \cancel{2} \cancel{3} \cancel{4} \cancel{5} \cancel{6} \cancel{7} \cancel{8} \cancel{9} \cancel{10}$$

$$ep = \cancel{2} \cancel{3} \cancel{4} \cancel{5} \cancel{6} \cancel{7} \cancel{8} \cancel{9} \cancel{10}$$

$$M(3)(1) = 1 \quad M(1)(0) = 1$$

$$3 \rightarrow 1$$



$$\text{int } a=0$$