

## Problem Statement

### Maximum Nesting Depth of Parentheses

Given a valid parentheses string  $s$ , return the **maximum nesting depth** of the parentheses.

The **nesting depth** is the maximum number of '(' that are open at the same time.

---

### Example

Input:  $s = "(1+(2*3)+((8)/4))+1"$

Output: 3

Explanation:

- Deepest nesting is  $"((8)/4)" \rightarrow \text{depth} = 3$
- 

### Approach (Simple & Intuitive)

We **don't need a stack**.

**Key idea:**

- Increase depth when you see '('
  - Decrease depth when you see ')'
  - Track the **maximum depth** at any point
- 

### Java Code (Counter-Based, $O(n)$ )

```
class Solution {  
    public int maxDepth(String s) {  
  
        int currDepth = 0;  
        int maxDepth = 0;  
  
        for (int i = 0; i < s.length(); i++) {  
            char ch = s.charAt(i);  
  
            if (ch == '(') {  
                currDepth++;  
                maxDepth = Math.max(maxDepth, currDepth);  
            }  
        }  
    }  
}
```

```

    }
    else if (ch == ')') {
        currDepth--;
    }
}

return maxDepth;
}
}

```

### Dry Run (Step-by-Step)

**Input:**

`s = "(1+(2*3)+((8)/4))+1"`

#### Tracking Variables

**Character currDepth maxDepth**

(	1	1
1	1	1
+	1	1
(	2	2
2	2	2
*	2	2
3	2	2
)	1	2
+	1	2
(	2	2
(	3	3
8	3	3
)	2	3

Character	currDepth	maxDepth
-----------	-----------	----------

/	2	3
---	---	---

4	2	3
---	---	---

)	1	3
---	---	---

)	0	3
---	---	---

+	0	3
---	---	---

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

---

 **Final Answer**

3

---

 **Time Complexity**

Operation	Complexity
-----------	------------

Single traversal of string	$O(n)$
----------------------------	--------

<b>Total Time</b>	<b><math>O(n)</math></b>
-------------------	--------------------------

---

 **Space Complexity**

Structure	Space
-----------	-------

Variables only	$O(1)$
----------------	--------

<b>Total Space</b>	<b><math>O(1)</math></b>
--------------------	--------------------------