678. Valid Parenthesis String

1. 678. Valid Parenthesis String

Medium 3000 74 Add to List Share

Given a string `s` containing only three types of characters: `'('`, `')'` and `'*'`, return `true` *if* `s` *is* ***valid***.

The following rules define a **valid** string:

- Any left parenthesis `'('` must have a corresponding right parenthesis `')'`.
- Any right parenthesis `')'` must have a corresponding left parenthesis `'('`.
- Left parenthesis `'('` must go before the corresponding right parenthesis `')'`.
- `'*'` could be treated as a single right parenthesis `')'` or a single left parenthesis `'('` or an empty string `""`.

Example 1:

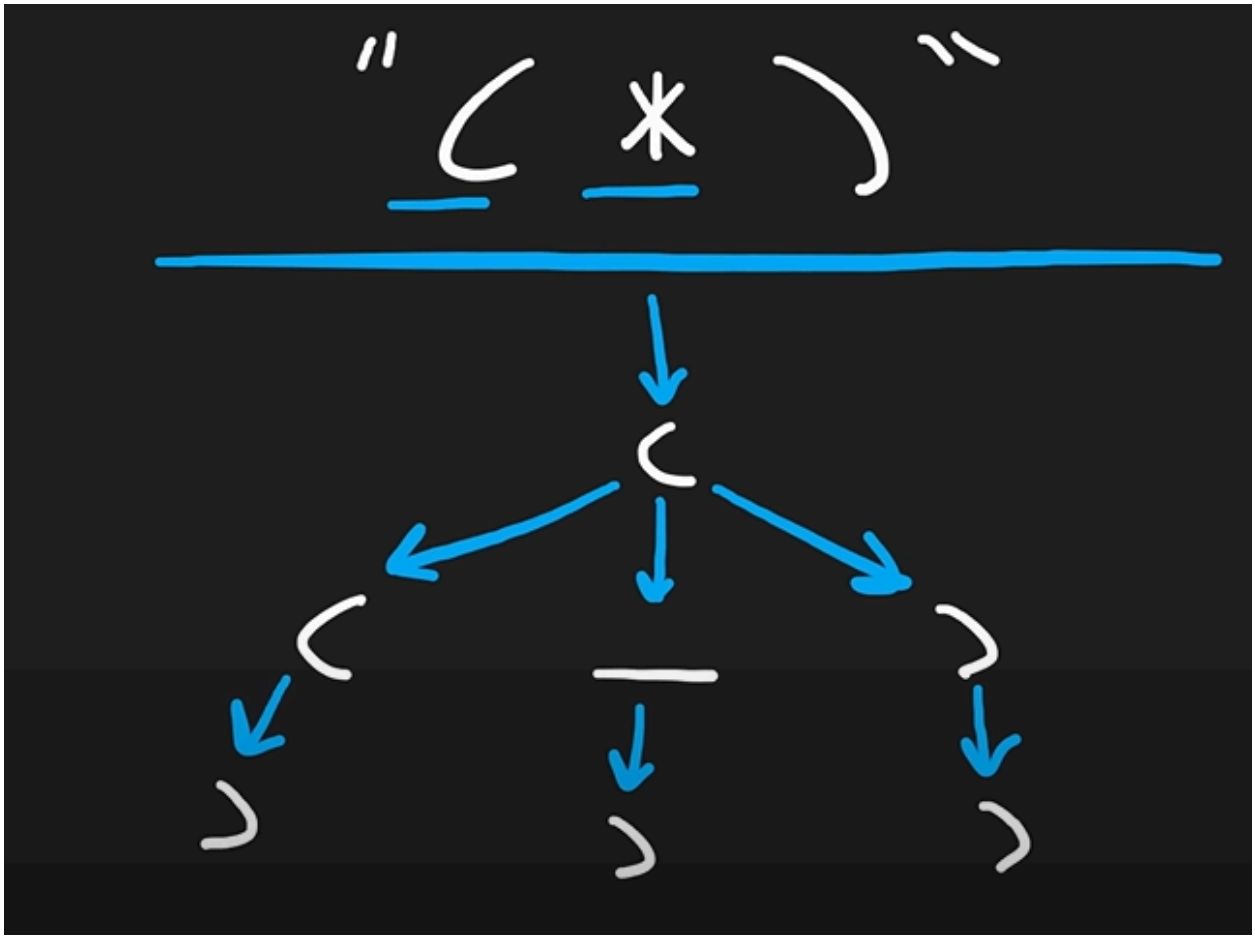
```
Input: s = "()"
Output: true
```

Example 2:

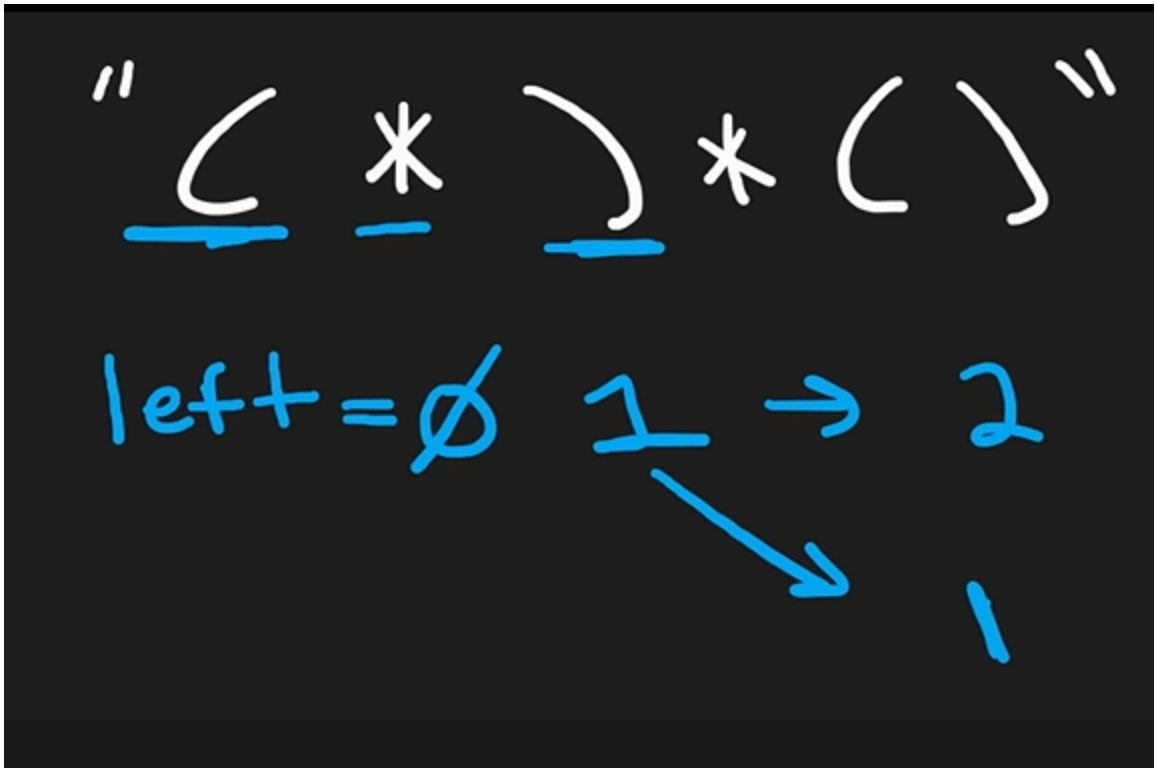
```
Input: s = "(*)"
Output: true
```

2. Invalid example : 

3. Lets take another example



4. Here '*' can mean either '(', '_',')'...so we created a decision tree…and at end we have append our last char ')' to every node in the tree
5. We can solve this by Dynamic programming(memoization)...but it takes O(n^3) time complexity(see vid @5:00)
6. So we can solve this in greedy approach too
7. Here we'll be using the left variable to store the count of 'open brackets' if we encounter '(' we increment it and if ')' then decrement left



here if first we have encountered '(' then we incremented it and '*' this can mean 2 things..so count has to be adjusted

8. So here we will be using two variables Leftmin and leftmax ..which is used to store the possibilities of '*'

Leetcode Explanation :

1. One way to approach this problem is to use a greedy strategy. We can keep track of the minimum and maximum number of open parentheses that must be matched and see if it's possible to match all the parentheses in the string. We'll use two variables `leftMin` and `leftMax` to represent the minimum and maximum number of open parentheses respectively.

Approach

1. Initialize `leftMin` and `leftMax` to 0.

2. Iterate through each character in the string `s`.

3. If the character is '(', increment both `leftMin` and `leftMax` by 1.

4. If the character is ')', decrement both `leftMin` and `leftMax` by 1.

5. If the character is '*', decrement `leftMin` by 1 and increment `leftMax` by 1.

6. If `leftMax` becomes negative at any point, return False since it means there are more closing parentheses than opening ones.

7. If `leftMin` becomes negative, reset it to 0 since we can't have negative open parentheses count.

8. After iterating through the string, check if `leftMin` is 0. If it is, return True; otherwise, return False.

- ## Time complexity:

$O(n)$, where n is the length of the input string `s`. We iterate through the string once.

- ## Space complexity:

$O(1)$, as we only use a constant amount of extra space for variables.
