

1.) Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`. You may assume that each input would have exactly one solution, and you may not use the same element twice.

Sent by you: Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`. You may assume that each input would have exactly one solution, and you may not use the same element twice.

Input: `nums = [2,7,11,15]`, `target = 9` Output: `[0,1]` Explanation: Because `nums[0] + nums[1] == 9`, we return `[0, 1]`.

Sent by you: Input: `nums = [2,7,11,15]`, `target = 9` Output: `[0,1]` Explanation: Because `nums[0] + nums[1] == 9`, we return `[0, 1]`.

The screenshot displays the LeetCode submission page for the 'Two Sum' problem. The submission is marked as 'Accepted' and was submitted by 'Surya Narayanan' on June 03, 2024, at 22:52. The performance metrics show a runtime of 57 ms, which beats 73.00% of users with Python3, and a memory usage of 17.67 MB, which beats 54.99% of users with Python3. A bar chart illustrates the runtime distribution across various time intervals. The code editor on the right shows the following Python3 solution:

```
1 class Solution:
2     def twoSum(self, nums: List[int], target: int) -> List[int]:
3         prevMap = {}
4
5         for i, n in enumerate(nums):
6             diff = target - n
7             if diff in prevMap:
8                 return [prevMap[diff], i]
9             prevMap[n] = i
```

Below the code editor, the 'Testcase' section shows the input: `nums = [2, 7, 11, 15]` and `target = 9`. The 'Test Result' section shows the output: `[0, 1]`.

Time :  $O(n)$

Space:  $O(n)$

2.) **Add Two Numbers** You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list. You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1: Input: l1 = [2,4,3], l2 = [5,6,4] Output: [7,0,8] Explanation: 342 + 465 = 807.

The screenshot displays the LeetCode submission interface for the "Add Two Numbers" problem. The top section shows the problem description and submission statistics: Runtime is 52 ms, beating 72.70% of users with Python3, and Memory is 16.70 MB, beating 43.22% of users with Python3. A runtime graph shows the distribution of execution times. The code editor on the right contains a Python solution for the problem. The bottom section shows the test result, which is "Accepted" with a runtime of 46 ms. The input is l1 = [2,4,3] and l2 = [5,6,4], and the output is [7,0,8], matching the expected result.

```
class Solution:
    def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNode:
        dummyHead = ListNode(0)
        tail = dummyHead
        carry = 0

        while l1 is not None or l2 is not None or carry != 0:
            d1 = l1.val if l1 is not None else 0
            d2 = l2.val if l2 is not None else 0

            sum = d1 + d2 + carry
            digit = sum % 10
            carry = sum // 10

            newNode = ListNode(digit)
            tail.next = newNode
            tail = tail.next

            l1 = l1.next if l1 is not None else None
            l2 = l2.next if l2 is not None else None

        result = dummyHead.next
        dummyHead.next = None
        return result
```

**Testcase** | **Test Result**

**Accepted** Runtime: 46 ms

• Case 1 • Case 2 • Case 3

Input

l1 =  
[2,4,3]

l2 =  
[5,6,4]

Output

[7,0,8]

Expected

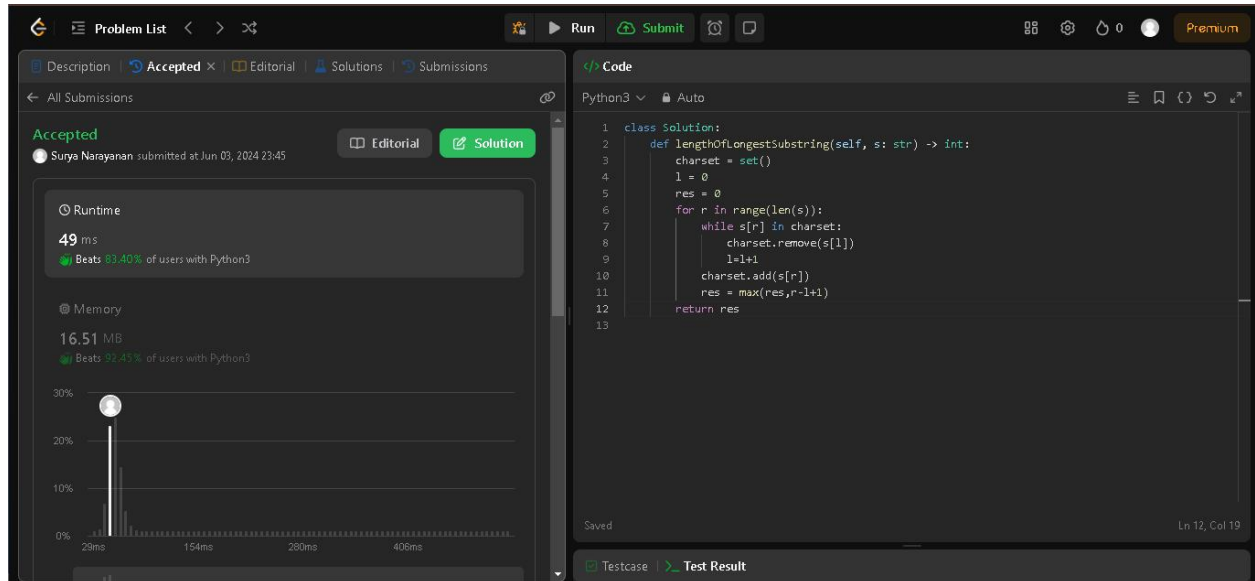
[7,0,8]

Time :  $O(\max(m,n))$

3. **Longest Substring without Repeating Characters** Given a string s, find the length of the longest substring without repeating characters.

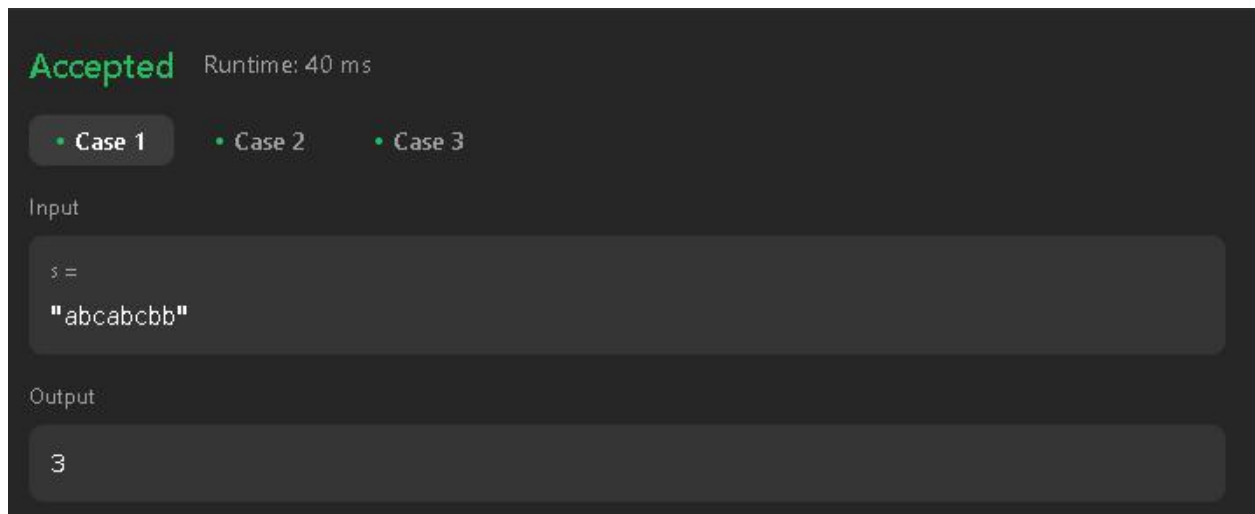
Example 1: Input: s = "abcabcbb" Output: 3 Explanation: The answer is "abc", with the length of 3.

Example 2: Input: s = "bbbbbb" Output: 1 Explanation: The answer is "b", with the length of 1



The screenshot shows a LeetCode submission interface. On the left, the 'Accepted' status is confirmed, with a runtime of 49 ms and memory usage of 16.51 MB. The right panel displays the Python code for the solution:

```
1 class Solution:
2     def lengthOfLongestSubstring(self, s: str) -> int:
3         charset = set()
4         l = 0
5         res = 0
6         for r in range(len(s)):
7             while s[r] in charset:
8                 charset.remove(s[l])
9                 l += 1
10            charset.add(s[r])
11            res = max(res, r - l + 1)
12        return res
13
```



The screenshot shows the test case results for the 'Longest Substring Without Repeating Characters' problem. The status is 'Accepted' with a runtime of 40 ms. The input is s = "abcabcbb" and the output is 3.

Accepted Runtime: 40 ms

- Case 1
- Case 2
- Case 3

Input

s = "abcabcbb"

Output

3

Time:  $O(n)$

Space:  $O(n)$

**4. Median of Two Sorted Arrays** Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays. The overall run time complexity should be  $O(\log(m+n))$ .

Example 1: Input: nums1 = [1,3], nums2 = [2] Output: 2.00000 Explanation: merged array = [1,2,3] and median is 2.

Example 2: Input: nums1 = [1,2], nums2 = [3,4] Output: 2.50000 Explanation: merged array = [1,2,3,4] and median is  $(2 + 3) / 2 = 2.5$

The screenshot displays a LeetCode submission page for the problem "Find Median Sorted Arrays". The submission is marked as "Accepted" and was submitted by "Surya Narayanan" on June 04, 2024, at 00:43. The runtime is 80 ms, which beats 60.73% of users with Python3. The memory usage is 16.88 MB, which beats 76.30% of users with Python3. A performance graph shows the distribution of runtimes, with the submission marked as a "1" on the curve.

The code editor shows the following Python3 solution:

```
1 class Solution:
2     def findMedianSortedArrays(self, nums1, nums2):
3         n = len(nums1)
4         m = len(nums2)
5         i = 0
6         j = 0
7         m1 = 0
8         m2 = 0
9         for count in range(0, (n + m) // 2 + 1):
10             m2 = m1
11             if i < n and j < m:
12                 if nums1[i] > nums2[j]:
13                     m1 = nums2[j]
14                     j += 1
15                 else:
16                     m1 = nums1[i]
17                     i += 1
18             elif i < n:
19                 m1 = nums1[i]
20                 i += 1
21             else:
22                 m1 = nums2[j]
23                 j += 1
```

The test result section shows the input and output for a specific test case:

Input:

```
nums1 = [1,3]
nums2 = [2]
```

Output:

```
2.00000
```

5.) **Longest Palindromic Substring** Given a string  $s$ , return the longest palindromic substring in  $s$ .

Example 1: Input:  $s = \text{"babad"}$  Output:  $\text{"bab"}$  Explanation:  $\text{"aba"}$  is also a valid answer.

Example 2: Input:  $s = \text{"cbbd"}$  Output:  $\text{"bb"}$

```
1 class Solution:
2     def longestPalindrome(self, s: str) -> str:
3         if len(s) <= 1:
4             return s
5         def expand_from_center(left, right):
6             while left >= 0 and right < len(s) and s[left] == s[right]:
7                 left -= 1
8                 right += 1
9             return s[left + 1:right]
10        max_str = s[0]
11        for i in range(len(s) - 1):
12            odd = expand_from_center(i, i)
13            even = expand_from_center(i, i + 1)
14
15            if len(odd) > len(max_str):
16                max_str = odd
17            if len(even) > len(max_str):
18                max_str = even
19        return max_str
```

Input

$s =$

$\text{"babad"}$

Output

$\text{"bab"}$

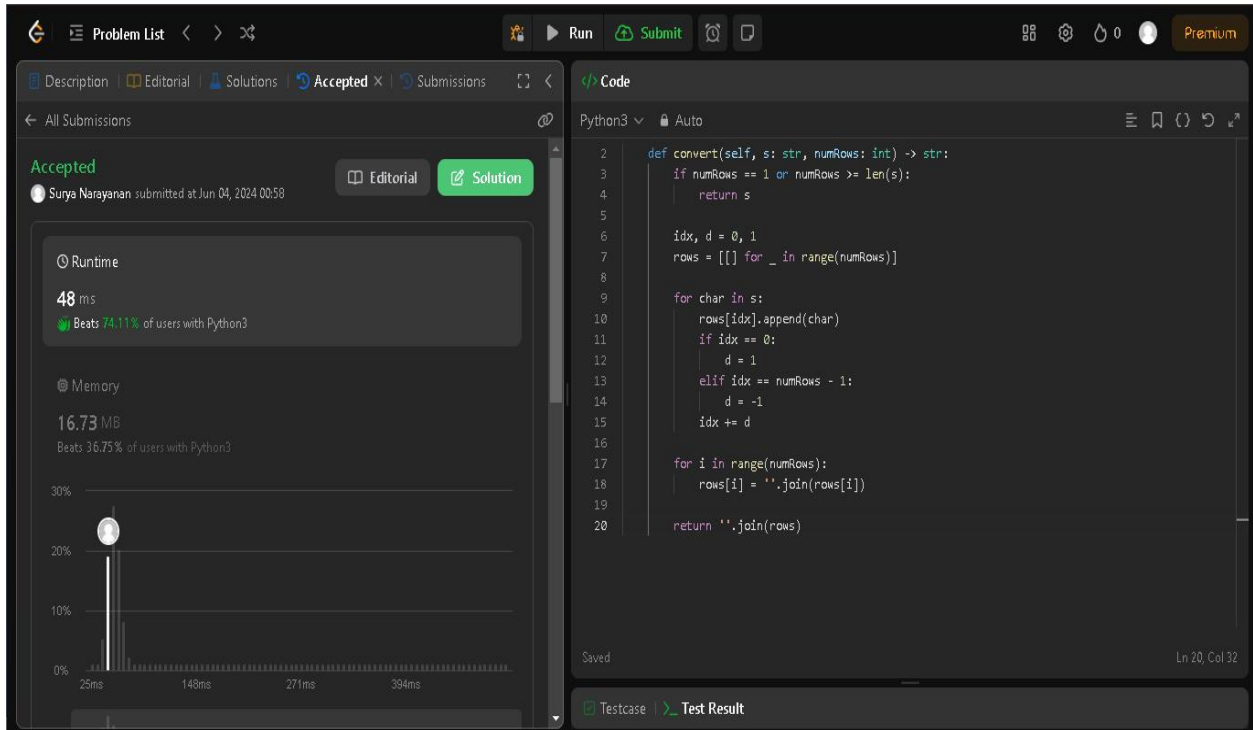
Time:  $O(n^2)$

Space:  $O(1)$

**6. Zigzag Conversion** The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)  
P A H N A P L S I I G Y I R  
And then read line by line: "PAHNAPLSIIGYIR" Write the code that will take a string and make this conversion given a number of rows: string convert(string s, int numRows);

Example 1: Input: s = "PAYPALISHIRING", numRows = 3 Output: "PAHNAPLSIIGYIR"

Example 2: Input: s = "PAYPALISHIRING", numRows = 4 Output: "PINALSIGYAHRPI"



The screenshot shows a LeetCode submission interface. On the left, the 'Accepted' status is confirmed, with a runtime of 48 ms and memory usage of 16.73 MB. The main area displays the Python code for the solution:

```
def convert(self, s: str, numRows: int) -> str:
    if numRows == 1 or numRows >= len(s):
        return s

    idx, d = 0, 1
    rows = [[] for _ in range(numRows)]

    for char in s:
        rows[idx].append(char)
        if idx == 0:
            d = 1
        elif idx == numRows - 1:
            d = -1
        idx += d

    for i in range(numRows):
        rows[i] = ''.join(rows[i])

    return ''.join(rows)
```

Input

s =  
"PAYPALISHIRING"

numRows =  
3

Output

"PAHNAPLSIIGYIR"

Time:  $O(n)$

Space:  $O(n)$

**7. Reverse Integer** Given a signed 32-bit integer  $x$ , return  $x$  with its digits reversed. If reversing  $x$  causes the value to go outside the signed 32-bit integer range  $[-2^{31}, 2^{31} - 1]$ , then return 0. Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1: Input:  $x = 123$  Output: 321

Example 2: Input:  $x = -123$  Output: -321

The screenshot displays the LeetCode submission interface for the 'Reverse Integer' problem. The top navigation bar includes 'Problem List', 'Run', 'Submit', and 'Premium' buttons. The left sidebar shows the 'Accepted' status and a bar chart comparing the solution's performance to other users. The main area contains the code editor and the test result.

**Accepted**  
Surya Narayanan submitted at Jun 04, 2024 01:02

**Runtime**  
36 ms  
Beats 99.79% of users with Python3

**Memory**  
16.50 MB  
Beats 61.91% of users with Python3

**Code**

```
1 class Solution:
2     def reverse(self, x: int) -> int:
3         res = 0
4         if x < 0:
5             res = int(str(x)[1:][::-1]) * -1
6         else:
7             res = int(str(x)[::-1])
8
9         if res > 2 ** 31 - 1 or res < -2 ** 31:
10             return 0
11
12         return res
```

**Testcase** | **Test Result**

**Accepted** Runtime: 41 ms

The screenshot shows the input and output fields for the 'Reverse Integer' problem. The input field is labeled 'Input' and contains 'x = 123'. The output field is labeled 'Output' and contains '321'. Below the output field is a section labeled 'Expected'.

**Input**

x =  
123

**Output**

321

**Expected**

Time:  $O(n)$

Space:  $O(n)$

**8. String to Integer (atoi)** Implement the `myAtoi(string s)` function, which converts a string to a 32-bit signed integer (similar to C/C++'s `atoi` function). The algorithm for `myAtoi(string s)` is as follows:

1. Read in and ignore any leading whitespace.
2. Check if the next character (if not already at the end of the string) is '-' or '+'. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
3. Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.
4. Convert these digits into an integer (i.e. "123" -> 123, "0032" -> 32). If no digits were read, then the integer is 0. Change the sign as necessary (from step 2).
5. If the integer is out of the 32-bit signed integer range  $[-2^{31}, 2^{31} - 1]$ , then clamp the integer so that it remains in the range. Specifically, integers less than  $-2^{31}$  should be clamped to  $-2^{31}$ , and integers greater than  $2^{31} - 1$  should be clamped to  $2^{31} - 1$ .
6. Return the integer as the final result



Problem List

Run
Submit

Premium

Description
Editorial
Solutions
Accepted X
Submissions

All Submissions

Accepted

Surya Narayanan submitted at Jun 04, 2024 01:07

Editorial
Solution

Runtime
32 ms
Beats 88.02% of users with Python3

Memory
16.30 MB
Beats 99.64% of users with Python3

10%
5%
0%
15ms 21ms 28ms 31ms 36ms 41ms 46ms

Code
Python3
Auto

```

1 class Solution(object):
2     def myAtoi(self, s):
3         num = '0123456789'
4         res = ''
5         for x in s:
6             if x == ' ' and len(res) == 0:
7                 continue
8             if x != ' ' and (x in '+' or x in num) and len(res) == 0:
9                 res += x
10            elif x in num:
11                res += x
12            else:
13                break
14        if res == '' or res in '+-':
15            return 0
16        else:
17            if int(res) < -(2**31):
18                return -(2**31)
19            elif int(res) > (2**31 - 1):
20                return (2**31 - 1)
21            else:
22                return int(res)

```

Saved
Ln 22, Col 32

Testcase
Test Result

Input

s =  
"42"

Output

42

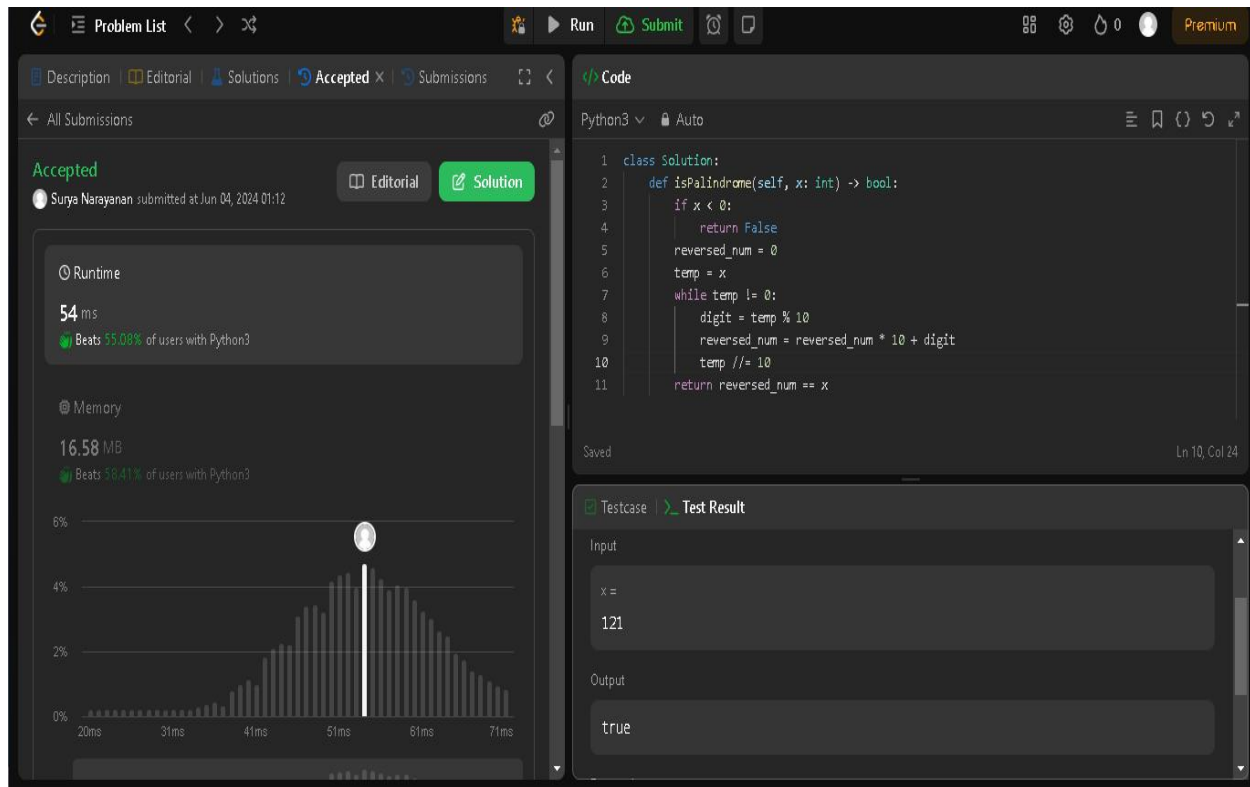
Time :  $O(n)$

Space :  $O(1)$

9.) Given an integer  $x$ , return true if  $x$  is a palindrome, and false otherwise.

Example 1: Input:  $x = 121$  Output: true Explanation: 121 reads as 121 from left to right and from right to left.

Example 2: Input:  $x = -121$  Output: false Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.



10.) **Regular Expression Matching** Given an input string  $s$  and a pattern  $p$ , implement regular expression matching with support for  $'.'$  and  $'*'$  where:

- $'.'$  Matches any single character.
- $'*'$  Matches zero or more of the preceding element.

The matching should cover the entire input string (not partial).

Example 1: Input:  $s = "aa"$ ,  $p = "a"$  Output: false Explanation: "a" does not match the entire string "aa".

Problem List

Accepted

Submissions

All Submissions

Accepted

Surya Narayanan submitted at Jun 04, 2024 01:15

Editorial

Solution

Runtime

40 ms

Beats 80.01% of users with Python3

Memory

16.60 MB

Beats 70.77% of users with Python3

0%

77ms

2435ms

4793ms

7151ms

100%

Code

Python3

Auto

```
1 class Solution:
2     def isMatch(self, s: str, p: str) -> bool:
3         i, j = len(s) - 1, len(p) - 1
4         return self.backtrack({}, s, p, i, j)
5
6     def backtrack(self, cache, s, p, i, j):
7         key = (i, j)
8         if key in cache:
9             return cache[key]
10
11         if i == -1 and j == -1:
12             cache[key] = True
13             return True
14
15         if i != -1 and j == -1:
16             cache[key] = False
17             return cache[key]
18
19         if i == -1 and p[j] == '*':
20             k = j
21             while k != -1 and p[k] == '*':
22                 k -= 2
23
```

Saved

Ln 51, Col 26

Testcase

Test Result

Problem List

Accepted

Submissions

All Submissions

Accepted

Surya Narayanan submitted at Jun 04, 2024 01:15

Editorial

Solution

Runtime

40 ms

Beats 80.01% of users with Python3

Memory

16.60 MB

Beats 70.77% of users with Python3

0%

77ms

2435ms

4793ms

7151ms

100%

Code

Python3

Auto

```
22         k -= 2
23
24         if k == -1:
25             cache[key] = True
26             return cache[key]
27
28         cache[key] = False
29         return cache[key]
30
31         if i == -1 and p[j] != '*':
32             cache[key] = False
33             return cache[key]
34
35         if p[j] == '*':
36             if self.backtrack(cache, s, p, i, j - 2):
37                 cache[key] = True
38                 return cache[key]
39
40             if p[j - 1] == s[i] or p[j - 1] == '.':
41                 if self.backtrack(cache, s, p, i - 1, j):
42                     cache[key] = True
43                     return cache[key]
44
45         if p[j] == '.' or s[i] == p[j]:
```

Saved

Ln 51, Col 26

Testcase

Test Result

```
Code
Python3 Auto
43         return cache[key]
44
45     if p[j] == '.' or s[i] == p[j]:
46         if self.backtrack(cache, s, p, i - 1, j - 1):
47             cache[key] = True
48             return cache[key]
49
50     cache[key] = False
51     return cache[key]
```

Saved Ln 51, Col 2

Testcase | Test Result

s =  
"aa"

p =  
"a"

Output  
false

Time : $O(m*n)$

Space: $O(m*n)$