Problem Challenge 1

We'll cover the following

Permutation in a String (hard)

Given a string and a pattern, find out if the **string contains any permutation of the pattern**.

Permutation is defined as the re-arranging of the characters of the string. For example, “abc” has the following six permutations:

1. abc
2. acb
3. bac
4. bca
5. cab
6. cba

If a string has ‘n’ distinct characters, it will have $n!$ permutations.

Example 1:

```
Input: String="oidbcaf", Pattern="abc"
Output: true
Explanation: The string contains "bca" which is a permutation of the given pattern.
```

Example 2:

```
Input: String="odicf", Pattern="dc"
Output: false
Explanation: No permutation of the pattern is present in the given string as a substring.
```

Example 3:

```
Input: String="bcdxabcdy", Pattern="bcdyabcdx"
Output: true
Explanation: Both the string and the pattern are a permutation of each other.
```

Example 4:

```
Input: String="aaacb", Pattern="abc"
Output: true
Explanation: The string contains "acb" which is a permutation of the given pattern.
```

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
def find_permutation(str, pattern):
  d = {}
  for c in pattern:
    if c in d:
```

```
    d[c] += 1
  else:
    d[c] = 1

p, q = 0, 0

while q < len(str):
  # We found a permutation
  if q - p == len(pattern):
    return True

  # The next char is not in pattern
  if str[q] not in d:
    while p < q:
      d[str[p]] += 1
      p += 1
    q += 1
    p += 1
  else:
    d[str[q]] -= 1
    q += 1

return q - p == len(pattern)

```

Test | Save | Reset

Back

Longest Subarray with Ones after Rep...

Next

Solution Review: Problem Challenge 1

Completed