

Problem 1216: Valid Palindrome III

Problem Information

Difficulty: **Hard**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given a string

`s`

and an integer

`k`

, return

`true`

if

`s`

is a

`k`

-palindrome

.

A string is

k

-palindrome

if it can be transformed into a palindrome by removing at most

k

characters from it.

Example 1:

Input:

s = "abcdeca", k = 2

Output:

true

Explanation:

Remove 'b' and 'e' characters.

Example 2:

Input:

s = "abbababa", k = 1

Output:

true

Constraints:

$1 \leq s.length \leq 1000$

s

consists of only lowercase English letters.

$1 \leq k \leq s.length$

Code Snippets

C++:

```
class Solution {
public:
    bool isValidPalindrome(string s, int k) {

    }
};
```

Java:

```
class Solution {
    public boolean isValidPalindrome(String s, int k) {

    }
}
```

Python3:

```
class Solution:
    def isValidPalindrome(self, s: str, k: int) -> bool:
```

Python:

```
class Solution(object):
    def isValidPalindrome(self, s, k):
        """
        :type s: str
        :type k: int
        :rtype: bool
        """
```

JavaScript:

```

/**
 * @param {string} s
 * @param {number} k
 * @return {boolean}
 */
var isValidPalindrome = function(s, k) {

};

```

TypeScript:

```

function isValidPalindrome(s: string, k: number): boolean {

};

```

C#:

```

public class Solution {
    public bool IsValidPalindrome(string s, int k) {

    }
}

```

C:

```

bool isValidPalindrome(char* s, int k) {

}

```

Go:

```

func isValidPalindrome(s string, k int) bool {

}

```

Kotlin:

```

class Solution {
    fun isValidPalindrome(s: String, k: Int): Boolean {

    }
}

```

Swift:

```
class Solution {  
    func isValidPalindrome(_ s: String, _ k: Int) -> Bool {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn is_valid_palindrome(s: String, k: i32) -> bool {  
  
    }  
}
```

Ruby:

```
# @param {String} s  
# @param {Integer} k  
# @return {Boolean}  
def is_valid_palindrome(s, k)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param String $s  
     * @param Integer $k  
     * @return Boolean  
     */  
    function isValidPalindrome($s, $k) {  
  
    }  
}
```

Dart:

```
class Solution {  
    bool isValidPalindrome(String s, int k) {  
  
    }  
}
```

```
}  
}
```

Scala:

```
object Solution {  
  def isValidPalindrome(s: String, k: Int): Boolean = {  
  
  }  
}
```

Elixir:

```
defmodule Solution do  
  @spec is_valid_palindrome(s :: String.t, k :: integer) :: boolean  
  def is_valid_palindrome(s, k) do  
  
  end  
end
```

Erlang:

```
-spec is_valid_palindrome(S :: unicode:unicode_binary(), K :: integer()) ->  
boolean().  
is_valid_palindrome(S, K) ->  
.
```

Racket:

```
(define/contract (is-valid-palindrome s k)  
  (-> string? exact-integer? boolean?)  
)
```

Solutions

C++ Solution:

```
/*  
 * Problem: Valid Palindrome III
```

```

* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
    bool isValidPalindrome(string s, int k) {

    }
};

```

Java Solution:

```

/**
 * Problem: Valid Palindrome III
 * Difficulty: Hard
 * Tags: string, dp
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

class Solution {
    public boolean isValidPalindrome(String s, int k) {

    }
}

```

Python3 Solution:

```

"""
Problem: Valid Palindrome III
Difficulty: Hard
Tags: string, dp

Approach: String manipulation with hash map or two pointers
"""

```

```

Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def isValidPalindrome(self, s: str, k: int) -> bool:
# TODO: Implement optimized solution
pass

```

Python Solution:

```

class Solution(object):
def isValidPalindrome(self, s, k):
"""
:type s: str
:type k: int
:rtype: bool
"""

```

JavaScript Solution:

```

/**
 * Problem: Valid Palindrome III
 * Difficulty: Hard
 * Tags: string, dp
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

/**
 * @param {string} s
 * @param {number} k
 * @return {boolean}
 */
var isValidPalindrome = function(s, k) {

};

```

TypeScript Solution:

```

/**
 * Problem: Valid Palindrome III
 * Difficulty: Hard
 * Tags: string, dp
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

function isValidPalindrome(s: string, k: number): boolean {

};

```

C# Solution:

```

/*
 * Problem: Valid Palindrome III
 * Difficulty: Hard
 * Tags: string, dp
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

public class Solution {
    public bool IsValidPalindrome(string s, int k) {

    }
}

```

C Solution:

```

/*
 * Problem: Valid Palindrome III
 * Difficulty: Hard
 * Tags: string, dp
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table

```

```

*/

bool isValidPalindrome(char* s, int k) {

}

```

Go Solution:

```

// Problem: Valid Palindrome III
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func isValidPalindrome(s string, k int) bool {

}

```

Kotlin Solution:

```

class Solution {
    fun isValidPalindrome(s: String, k: Int): Boolean {

    }
}

```

Swift Solution:

```

class Solution {
    func isValidPalindrome(_ s: String, _ k: Int) -> Bool {

    }
}

```

Rust Solution:

```

// Problem: Valid Palindrome III
// Difficulty: Hard
// Tags: string, dp

```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
    pub fn is_valid_palindrome(s: String, k: i32) -> bool {

    }
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} k
# @return {Boolean}
def is_valid_palindrome(s, k)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param String $s
     * @param Integer $k
     * @return Boolean
     */
    function isValidPalindrome($s, $k) {

    }

}
```

Dart Solution:

```
class Solution {
    bool isValidPalindrome(String s, int k) {

    }
}
```

Scala Solution:

```
object Solution {  
  def isValidPalindrome(s: String, k: Int): Boolean = {  
  
  }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec is_valid_palindrome(s :: String.t, k :: integer) :: boolean  
  def is_valid_palindrome(s, k) do  
  
  end  
end
```

Erlang Solution:

```
-spec is_valid_palindrome(S :: unicode:unicode_binary(), K :: integer()) ->  
boolean().  
is_valid_palindrome(S, K) ->  
.
```

Racket Solution:

```
(define/contract (is-valid-palindrome s k)  
  (-> string? exact-integer? boolean?)  
)
```