Problem 3445: Maximum Difference Between Even and Odd Frequency II

Problem Information

Difficulty: Hard
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

You are given a string

s

and an integer

k

. Your task is to find the

maximum

difference between the frequency of

two

characters,

freq[a] - freq[b]

, in a

substring

subs

of

s

, such that:

subs

has a size of

at least

k

.

Character

a

has an

odd frequency

in

subs

.

Character

b

has a

non-zero

even frequency

in

subs

.

Return the

maximum

difference.

Note

that

subs

can contain more than 2

distinct

characters.

Example 1:

Input:

s = "12233", k = 4

Output:

-1

Explanation:

For the substring

"12233"

, the frequency of

'1'

is 1 and the frequency of

'3'

is 2. The difference is

1 - 2 = -1

.

Example 2:

Input:

s = "1122211", k = 3

Output:

1

Explanation:

For the substring

"11222"

, the frequency of

'2'

is 3 and the frequency of

'1'

is 2. The difference is

3 - 2 = 1

.

Example 3:

Input:

s = "110", k = 3

Output:

-1

Constraints:

3 <= s.length <= 3 * 10

4

s

consists only of digits

'0'

to

'4'

.

The input is generated that at least one substring has a character with an even frequency and a character with an odd frequency.

1 <= k <= s.length

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function maxDifference(s: string, k: number): number {

};
```

C#:

```
public class Solution {
public int MaxDifference(string s, int k) {

}
}
```

C:

```
int maxDifference(char* s, int k) {

}
```

Go:

```
func maxDifference(s string, k int) int {

}
```

Kotlin:

```
class Solution {
fun maxDifference(s: String, k: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int maxDifference(String s, int k) {

}
}
```

Scala:

```
object Solution {
def maxDifference(s: String, k: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (max-difference s k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Difference Between Even and Odd Frequency II
* Difficulty: Hard
* Tags: array, string, tree
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Maximum Difference Between Even and Odd Frequency II
* Difficulty: Hard
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int maxDifference(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Difference Between Even and Odd Frequency II
Difficulty: Hard
Tags: array, string, tree

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
```

```
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Difference Between Even and Odd Frequency II
* Difficulty: Hard
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Difference Between Even and Odd Frequency II
* Difficulty: Hard
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function maxDifference(s: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Difference Between Even and Odd Frequency II
* Difficulty: Hard
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public int MaxDifference(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Difference Between Even and Odd Frequency II
* Difficulty: Hard
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
```

```
*/

int maxDifference(char* s, int k) {

}
```

Go Solution:

```
// Problem: Maximum Difference Between Even and Odd Frequency II
// Difficulty: Hard
// Tags: array, string, tree
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func maxDifference(s string, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxDifference(s: String, k: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Difference Between Even and Odd Frequency II
// Difficulty: Hard
// Tags: array, string, tree
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int maxDifference(String s, int k) {

}
}
```

Scala Solution:

```
object Solution {
def maxDifference(s: String, k: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (max-difference s k)
(-> string? exact-integer? exact-integer?)
)
```