Problem 3085: Minimum Deletions to Make String K-Special

Problem Information

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

Problem Description

You are given a string

word

and an integer

k

.

We consider

word

to be

k-special

if

|freq(word[i]) - freq(word[j])| <= k

for all indices

i

and

j

in the string.

Here,

freq(x)

denotes the

frequency

of the character

x

in

word

, and

|y|

denotes the absolute value of

y

.

Return

the

minimum

number of characters you need to delete to make

word

k-special

.

Example 1:

Input:

word = "aabcaba", k = 0

Output:

3

Explanation:

We can make

word

0

-special by deleting

2

occurrences of

"a"

and

1

occurrence of

"c"

. Therefore,

word

becomes equal to

"baba"

where

freq('a') == freq('b') == 2

.

Example 2:

Input:

word = "dabdcbdcdcd", k = 2

Output:

2

Explanation:

We can make

word

2

-special by deleting

1

occurrence of

"a"

and

1

occurrence of

"d"

. Therefore,

word

becomes equal to "bdcbdcdcd" where

freq('b') == 2

,

freq('c') == 3

, and

freq('d') == 4

.

Example 3:

Input:

word = "aaabaaa", k = 2

Output:

1

Explanation:

We can make

word

2

-special by deleting

1

occurrence of

"b"

. Therefore,

word

becomes equal to

"aaaaaa"

where each letter's frequency is now uniformly

6

.

Constraints:

1 <= word.length <= 10

5

0 <= k <= 10

5

word

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int minimumDeletions(string word, int k) {

}
};
```

Java:

```
class Solution {
public int minimumDeletions(String word, int k) {

}
}
```

Python3:

```
class Solution:
def minimumDeletions(self, word: str, k: int) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minimumDeletions(word: string, k: number): number {

};
```

C#:

```
public class Solution {
public int MinimumDeletions(string word, int k) {

}
}
```

C:

```
int minimumDeletions(char* word, int k) {

}
```

Go:

```
func minimumDeletions(word string, k int) int {

}
```

Kotlin:

```
class Solution {
fun minimumDeletions(word: String, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func minimumDeletions(_ word: String, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_deletions(word: String, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} word
# @param {Integer} k
# @return {Integer}
def minimum_deletions(word, k)

end
```

PHP:

```
class Solution {

/**
* @param String $word
* @param Integer $k
* @return Integer
*/
function minimumDeletions($word, $k) {

}
}
```

Dart:

```
class Solution {
int minimumDeletions(String word, int k) {
```

```
}
}
```

Scala:

```
object Solution {
def minimumDeletions(word: String, k: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec minimum_deletions(Word :: unicode:unicode_binary(), K :: integer()) ->
integer().
minimum_deletions(Word, K) ->
.
```

Racket:

```
(define/contract (minimum-deletions word k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Deletions to Make String K-Special
```

```
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int minimumDeletions(string word, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Deletions to Make String K-Special
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int minimumDeletions(String word, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Deletions to Make String K-Special
Difficulty: Medium
Tags: string, greedy, hash, sort

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

```
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Deletions to Make String K-Special
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Deletions to Make String K-Special
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function minimumDeletions(word: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Deletions to Make String K-Special
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int MinimumDeletions(string word, int k) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Deletions to Make String K-Special
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

int minimumDeletions(char* word, int k) {

}
```

Go Solution:

```
// Problem: Minimum Deletions to Make String K-Special
// Difficulty: Medium
// Tags: string, greedy, hash, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func minimumDeletions(word string, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumDeletions(word: String, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumDeletions(_ word: String, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Deletions to Make String K-Special
// Difficulty: Medium
// Tags: string, greedy, hash, sort
```

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

impl Solution {
pub fn minimum_deletions(word: String, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} word
# @param {Integer} k
# @return {Integer}
def minimum_deletions(word, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $word
* @param Integer $k
* @return Integer
*/
function minimumDeletions($word, $k) {

}
}
```

Dart Solution:

```
class Solution {
int minimumDeletions(String word, int k) {

}
}
```

Scala Solution:

```
object Solution {
def minimumDeletions(word: String, k: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec minimum_deletions(Word :: unicode:unicode_binary(), K :: integer()) ->
integer().
minimum_deletions(Word, K) ->
.
```

Racket Solution:

```
(define/contract (minimum-deletions word k)
(-> string? exact-integer? exact-integer?)
)
```