Problem 3545: Minimum Deletions for At Most K Distinct Characters

Problem Information

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

Problem Description

You are given a string

s

consisting of lowercase English letters, and an integer

k

.

Your task is to delete some (possibly none) of the characters in the string so that the number of

distinct

characters in the resulting string is

at most

k

.

Return the

minimum

number of deletions required to achieve this.

Example 1:

Input:

s = "abc", k = 2

Output:

1

Explanation:

s

has three distinct characters:

'a'

,

'b'

and

'c'

, each with a frequency of 1.

Since we can have at most

k = 2

distinct characters, remove all occurrences of any one character from the string.

For example, removing all occurrences of 'c'

results in at most

k

distinct characters. Thus, the answer is 1.

Example 2:

Input:

s = "aabb", k = 2

Output:

0

Explanation:

s

has two distinct characters (

'a'

and

'b'

) with frequencies of 2 and 2, respectively.

Since we can have at most

k = 2

distinct characters, no deletions are required. Thus, the answer is 0.

Example 3:

Input:

s = "yyyzz", k = 1

Output:

2

Explanation:

s

has two distinct characters (

'y'

and

'z'

) with frequencies of 3 and 2, respectively.

Since we can have at most

k = 1

distinct character, remove all occurrences of any one character from the string.

Removing all

'z'

results in at most

k

distinct characters. Thus, the answer is 2.

Constraints:

1 <= s.length <= 16

1 <= k <= 16

s

consists only of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def minDeletion(self, s, k):
"""
:type s: str
```

```
:type k: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (min-deletion s k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Deletions for At Most K Distinct Characters
* Difficulty: Easy
* 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 minDeletion(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Deletions for At Most K Distinct Characters
* Difficulty: Easy
* 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 minDeletion(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Deletions for At Most K Distinct Characters
```

```
Difficulty: Easy
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 minDeletion(self, s: str, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Deletions for At Most K Distinct Characters
* Difficulty: Easy
* 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} s
* @param {number} k
* @return {number}
*/
var minDeletion = function(s, k) {
};
```

TypeScript Solution:

```
/**
* Problem: Minimum Deletions for At Most K Distinct Characters
* Difficulty: Easy
* 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 minDeletion(s: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Deletions for At Most K Distinct Characters
* Difficulty: Easy
* 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 MinDeletion(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Deletions for At Most K Distinct Characters
* Difficulty: Easy
* 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 minDeletion(char* s, int k) {

}
```

Go Solution:

```
// Problem: Minimum Deletions for At Most K Distinct Characters
// Difficulty: Easy
// 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 minDeletion(s string, k int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Deletions for At Most K Distinct Characters
// Difficulty: Easy
// 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 min_deletion(s: String, k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (min-deletion s k)
(-> string? exact-integer? exact-integer?)
)
```