Problem 1309: Decrypt String from Alphabet to Integer Mapping

Problem Information

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

Problem Description

You are given a string

s

formed by digits and

'#'

. We want to map

s

to English lowercase characters as follows:

Characters (

'a'

to

'i'

) are represented by (

'1'

to

'9'

) respectively.

Characters (

'j'

to

'z'

) are represented by (

'10#'

to

'26#'

) respectively.

Return

the string formed after mapping

.

The test cases are generated so that a unique mapping will always exist.

Example 1:

Input:

s = "10#11#12"

Output:

"jkab"

Explanation:

"j" -> "10#" , "k" -> "11#" , "a" -> "1" , "b" -> "2".

Example 2:

Input:

s = "1326#"

Output:

"acz"

Constraints:

1 <= s.length <= 1000

s

consists of digits and the

'#'

letter.

s

will be a valid string such that mapping is always possible.

Code Snippets

C++:

```
class Solution {
public:
string freqAlphabets(string s) {

}
};
```

Java:

```
class Solution {
public String freqAlphabets(String s) {

}
}
```

Python3:

```
class Solution:
def freqAlphabets(self, s: str) -> str:
```

Python:

```
class Solution(object):
def freqAlphabets(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string}
*/
var freqAlphabets = function(s) {

};
```

TypeScript:

```
function freqAlphabets(s: string): string {
};
```

C#:

```
public class Solution {
public string FreqAlphabets(string s) {

}
}
```

C:

```
char* freqAlphabets(char* s) {

}
```

Go:

```
func freqAlphabets(s string) string {

}
```

Kotlin:

```
class Solution {
fun freqAlphabets(s: String): String {

}
}
```

Swift:

```
class Solution {
func freqAlphabets(_ s: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn freq_alphabets(s: String) -> String {
}
}
```

Ruby:

```
# @param {String} s
# @return {String}
def freq_alphabets(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String
*/
function freqAlphabets($s) {

}
}
```

Dart:

```
class Solution {
String freqAlphabets(String s) {

}
}
```

Scala:

```
object Solution {
def freqAlphabets(s: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec freq_alphabets(s :: String.t) :: String.t
def freq_alphabets(s) do

end
end
```

Erlang:

```
-spec freq_alphabets(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
freq_alphabets(S) ->
.
```

Racket:

```
(define/contract (freq-alphabets s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Decrypt String from Alphabet to Integer Mapping
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string freqAlphabets(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Decrypt String from Alphabet to Integer Mapping
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String freqAlphabets(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Decrypt String from Alphabet to Integer Mapping
Difficulty: Easy
Tags: string

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

```
class Solution(object):
def freqAlphabets(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Decrypt String from Alphabet to Integer Mapping
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @return {string}
*/
var freqAlphabets = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Decrypt String from Alphabet to Integer Mapping
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function freqAlphabets(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Decrypt String from Alphabet to Integer Mapping
* Difficulty: Easy
* Tags: string
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string FreqAlphabets(string s) {

}
}
```

C Solution:

```
/*
* Problem: Decrypt String from Alphabet to Integer Mapping
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* freqAlphabets(char* s) {

}
```

Go Solution:

```
// Problem: Decrypt String from Alphabet to Integer Mapping
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func freqAlphabets(s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun freqAlphabets(s: String): String {

}
}
```

Swift Solution:

```
class Solution {
func freqAlphabets(_ s: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Decrypt String from Alphabet to Integer Mapping
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn freq_alphabets(s: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String}
def freq_alphabets(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String
*/
function freqAlphabets($s) {

}
}
```

Dart Solution:

```
class Solution {
String freqAlphabets(String s) {

}
}
```

Scala Solution:

```
object Solution {
def freqAlphabets(s: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec freq_alphabets(s :: String.t) :: String.t
def freq_alphabets(s) do

end
end
```

Erlang Solution:

```
-spec freq_alphabets(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
freq_alphabets(S) ->
.
```

Racket Solution:

```
(define/contract (freq-alphabets s)
(-> string? string?)
)
```