Problem 1702: Maximum Binary String After Change

Problem Information

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

Problem Description

You are given a binary string

binary

consisting of only

0

's or

1

's. You can apply each of the following operations any number of times:

Operation 1: If the number contains the substring

"00"

, you can replace it with

"10"

.

For example,

"

00

010" -> "

10

010

"

Operation 2: If the number contains the substring

"10"

, you can replace it with

"01"

.

For example,

"000

10

" -> "000

01

"

Return the

maximum binary string

you can obtain after any number of operations. Binary string

x

is greater than binary string

y

if

x

's decimal representation is greater than

y

's decimal representation.

Example 1:

Input:

binary = "000110"

Output:

"111011"

Explanation:

A valid transformation sequence can be: "0001

10

" -> "0001

01

" "

00

0101" -> "

10

0101" "1

00

101" -> "1

10

101" "110

10

1" -> "110

01

1" "11

00

11" -> "11

10

11"

Example 2:

Input:

binary = "01"

Output:

"01"

Explanation:

"01" cannot be transformed any further.

Constraints:

1 <= binary.length <= 10

5

binary

consist of

'0'

and

'1'

.

Code Snippets

C++:

```
class Solution {
public:
string maximumBinaryString(string binary) {

}
};
```

Java:

```
class Solution {
public String maximumBinaryString(String binary) {

}
}
```

Python3:

```
class Solution:
def maximumBinaryString(self, binary: str) -> str:
```

Python:

```
class Solution(object):
def maximumBinaryString(self, binary):
"""
:type binary: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} binary
* @return {string}
*/
var maximumBinaryString = function(binary) {

};
```

TypeScript:

```
function maximumBinaryString(binary: string): string {

};
```

C#:

```
public class Solution {
public string MaximumBinaryString(string binary) {

}
}
```

C:

```
char* maximumBinaryString(char* binary) {

}
```

Go:

```
func maximumBinaryString(binary string) string {

}
```

Kotlin:

```
class Solution {
fun maximumBinaryString(binary: String): String {

}
}
```

Swift:

```
class Solution {
func maximumBinaryString(_ binary: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_binary_string(binary: String) -> String {

}
}
```

Ruby:

```
# @param {String} binary
# @return {String}
def maximum_binary_string(binary)

end
```

PHP:

```
class Solution {

/**
* @param String $binary
* @return String
*/
function maximumBinaryString($binary) {

}
}
```

Dart:

```
class Solution {
String maximumBinaryString(String binary) {

}
}
```

Scala:

```
object Solution {
def maximumBinaryString(binary: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximum_binary_string(binary :: String.t) :: String.t
def maximum_binary_string(binary) do

end
end
```

Erlang:

```
-spec maximum_binary_string(Binary :: unicode:unicode_binary()) ->
unicode:unicode_binary().
maximum_binary_string(Binary) ->
```

```
.
```

Racket:

```
(define/contract (maximum-binary-string binary)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Binary String After Change
* Difficulty: Medium
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
string maximumBinaryString(string binary) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Binary String After Change
* Difficulty: Medium
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
class Solution {
public String maximumBinaryString(String binary) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Binary String After Change
Difficulty: Medium
Tags: string, tree, greedy

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

```
class Solution(object):
def maximumBinaryString(self, binary):
"""
:type binary: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Binary String After Change
* Difficulty: Medium
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {string} binary
* @return {string}
*/
var maximumBinaryString = function(binary) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Binary String After Change
* Difficulty: Medium
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function maximumBinaryString(binary: string): string {

};
```

C# Solution:

```
/*
* Problem: Maximum Binary String After Change
* Difficulty: Medium
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
```

```
public string MaximumBinaryString(string binary) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Binary String After Change
* Difficulty: Medium
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

char* maximumBinaryString(char* binary) {

}
```

Go Solution:

```
// Problem: Maximum Binary String After Change
// Difficulty: Medium
// Tags: string, tree, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func maximumBinaryString(binary string) string {

}
```

Kotlin Solution:

```
class Solution {
fun maximumBinaryString(binary: String): String {

}
```

```
}
```

Swift Solution:

```
class Solution {
func maximumBinaryString(_ binary: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Maximum Binary String After Change
// Difficulty: Medium
// Tags: string, tree, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn maximum_binary_string(binary: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} binary
# @return {String}
def maximum_binary_string(binary)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $binary
* @return String
```

```
*/
function maximumBinaryString($binary) {

}
}
```

Dart Solution:

```
class Solution {
String maximumBinaryString(String binary) {

}
}
```

Scala Solution:

```
object Solution {
def maximumBinaryString(binary: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_binary_string(binary :: String.t) :: String.t
def maximum_binary_string(binary) do

end
end
```

Erlang Solution:

```
-spec maximum_binary_string(Binary :: unicode:unicode_binary()) ->
unicode:unicode_binary().
maximum_binary_string(Binary) ->
.
```

Racket Solution:

```
(define/contract (maximum-binary-string binary)
(-> string? string?)
)
```