Problem 2864: Maximum Odd Binary Number

Problem Information

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

Problem Description

You are given a

binary

string

s

that contains at least one

'1'

.

You have to

rearrange

the bits in such a way that the resulting binary number is the

maximum odd binary number

that can be created from this combination.

Return

a string representing the maximum odd binary number that can be created from the given combination.

Note

that the resulting string

can

have leading zeros.

Example 1:

Input:

s = "010"

Output:

"001"

Explanation:

Because there is just one '1', it must be in the last position. So the answer is "001".

Example 2:

Input:

s = "0101"

Output:

"1001"

Explanation:

One of the '1's must be in the last position. The maximum number that can be made with the remaining digits is "100". So the answer is "1001".

Constraints:

1 <= s.length <= 100

s

consists only of

'0'

and

'1'

.

s

contains at least one

'1'

.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public String maximumOddBinaryNumber(String s) {
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function maximumOddBinaryNumber(s: string): string {

};
```

C#:

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

}
}
```

C:

```
char* maximumOddBinaryNumber(char* s) {

}
```

Go:

```
func maximumOddBinaryNumber(s string) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

```
.
```

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Maximum Odd Binary Number
* Difficulty: Easy
* Tags: string, greedy, math
*
* 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 maximumOddBinaryNumber(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Odd Binary Number
* Difficulty: Easy
* Tags: string, greedy, math
*
* 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 maximumOddBinaryNumber(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Odd Binary Number
Difficulty: Easy
Tags: string, greedy, math

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 maximumOddBinaryNumber(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Odd Binary Number
* Difficulty: Easy
* Tags: string, greedy, math
*
* 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 maximumOddBinaryNumber = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Odd Binary Number
* Difficulty: Easy
* Tags: string, greedy, math
*
* 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 maximumOddBinaryNumber(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Maximum Odd Binary Number
* Difficulty: Easy
* Tags: string, greedy, math
*
* 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 MaximumOddBinaryNumber(string s) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Odd Binary Number
* Difficulty: Easy
* Tags: string, greedy, math
*
* 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* maximumOddBinaryNumber(char* s) {

}
```

Go Solution:

```
// Problem: Maximum Odd Binary Number
// Difficulty: Easy
// Tags: string, greedy, math
//
// 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 maximumOddBinaryNumber(s string) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Odd Binary Number
// Difficulty: Easy
// Tags: string, greedy, math
//
// 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 maximum_odd_binary_number(s: String) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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