# Problem 1529: Minimum Suffix Flips

## Problem Information

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

## Problem Description

You are given a

0-indexed

binary string

target

of length

n

. You have another binary string

s

of length

n

that is initially set to all zeros. You want to make

s

equal to

target

.

In one operation, you can pick an index

i

where

0 <= i < n

and flip all bits in the

inclusive

range

[i, n - 1]

. Flip means changing

'0'

to

'1'

and

'1'

to

'0'

.

Return

the minimum number of operations needed to make

s

equal to

target

.

Example 1:

Input:

target = "10111"

Output:

3

Explanation:

Initially, s = "00000". Choose index i = 2: "00

000

" -> "00

111

" Choose index i = 0: "

00111

" -> "

11000

" Choose index i = 1: "1

1000

" -> "1

0111

" We need at least 3 flip operations to form target.

Example 2:

Input:

target = "101"

Output:

3

Explanation:

Initially, s = "000". Choose index i = 0: "

000

" -> "

111

" Choose index i = 1: "1

11

" -> "1

00

" Choose index i = 2: "10

0

" -> "10

1

" We need at least 3 flip operations to form target.

Example 3:

Input:

target = "00000"

Output:

0

Explanation:

We do not need any operations since the initial s already equals target.

Constraints:

n == target.length

1 <= n <= 10

5

target[i]

is either

'0'

or

'1'

.

## Code Snippets

### C++:

```
class Solution {
public:
int minFlips(string target) {

}
};
```

### Java:

```
class Solution {
public int minFlips(String target) {

}
}
```

### Python3:

```
class Solution:
def minFlips(self, target: str) -> int:
```

### Python:

```
class Solution(object):
def minFlips(self, target):
"""
:type target: str
:rtype: int
"""
```

### JavaScript:

```
/**
* @param {string} target
* @return {number}
*/
var minFlips = function(target) {

};
```

### TypeScript:

```
function minFlips(target: string): number {

};
```

### C#:

```
public class Solution {
public int MinFlips(string target) {

}
}
```

### C:

```
int minFlips(char* target) {

}
```

### Go:

```
func minFlips(target string) int {

}
```

### Kotlin:

```
class Solution {
fun minFlips(target: String): Int {

}
}
```

### Swift:

```
class Solution {
func minFlips(_ target: String) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn min_flips(target: String) -> i32 {

}
}
```

**Ruby:**

```
# @param {String} target
# @return {Integer}
def min_flips(target)

end
```

**PHP:**

```
class Solution {

/**
* @param String $target
* @return Integer
*/
function minFlips($target) {

}
}
```

**Dart:**

```
class Solution {
int minFlips(String target) {

}
}
```

### Scala:

```
object Solution {
def minFlips(target: String): Int = {

}
}
```

### Elixir:

```
defmodule Solution do
@spec min_flips(target :: String.t) :: integer
def min_flips(target) do

end
end
```

### Erlang:

```
-spec min_flips(Target :: unicode:unicode_binary()) -> integer().
min_flips(Target) ->
.
```

### Racket:

```
(define/contract (min-flips target)
(-> string? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Minimum Suffix Flips
* Difficulty: Medium
* Tags: string, greedy
*
* 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:
int minFlips(string target) {

}
};
```

### Java Solution:

```
/**
* Problem: Minimum Suffix Flips
* Difficulty: Medium
* Tags: string, greedy
*
* 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 int minFlips(String target) {

}
}
```

### Python3 Solution:

```
"""
Problem: Minimum Suffix Flips
Difficulty: Medium
Tags: string, greedy

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 minFlips(self, target: str) -> int:
# TODO: Implement optimized solution
```

```
pass
```

### Python Solution:

```
class Solution(object):
def minFlips(self, target):
"""
:type target: str
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Minimum Suffix Flips
* Difficulty: Medium
* Tags: string, greedy
*
* 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} target
* @return {number}
*/
var minFlips = function(target) {

};
```

### TypeScript Solution:

```
/**
* Problem: Minimum Suffix Flips
* Difficulty: Medium
* Tags: string, greedy
*
* 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 minFlips(target: string): number {

};
```

### C# Solution:

```
/*
* Problem: Minimum Suffix Flips
* Difficulty: Medium
* Tags: string, greedy
*
* 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 int MinFlips(string target) {

}
}
```

### C Solution:

```
/*
* Problem: Minimum Suffix Flips
* Difficulty: Medium
* Tags: string, greedy
*
* 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
*/

int minFlips(char* target) {

}
```

### Go Solution:

```
// Problem: Minimum Suffix Flips
// Difficulty: Medium
// Tags: string, greedy
//
// 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 minFlips(target string) int {

}
```

### Kotlin Solution:

```
class Solution {
fun minFlips(target: String): Int {

}
}
```

### Swift Solution:

```
class Solution {
func minFlips(_ target: String) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Minimum Suffix Flips
// Difficulty: Medium
// Tags: string, greedy
//
// 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 min_flips(target: String) -> i32 {

}
}
```

### Ruby Solution:

```
# @param {String} target
# @return {Integer}
def min_flips(target)

end
```

### PHP Solution:

```
class Solution {

/**
* @param String $target
* @return Integer
*/
function minFlips($target) {

}
}
```

### Dart Solution:

```
class Solution {
int minFlips(String target) {

}
}
```

### Scala Solution:

```
object Solution {
def minFlips(target: String): Int = {

}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec min_flips(target :: String.t) :: integer
def min_flips(target) do

end
end
```

## Erlang Solution:

```
-spec min_flips(Target :: unicode:unicode_binary()) -> integer().
min_flips(Target) ->
.
```

## Racket Solution:

```
(define/contract (min-flips target)
(-> string? exact-integer?)
)
```