Problem 13: Roman to Integer

Problem Information

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

Problem Description

Roman numerals are represented by seven different symbols:

I

,

V

,

X

,

L

,

C

,

D

and

M

.

Symbol

Value

I 1 V 5 X 10 L 50 C 100 D 500 M 1000

For example,

2

is written as

II

in Roman numeral, just two ones added together.

12

is written as

XII

, which is simply

X + II

. The number

27

is written as

XXVII

, which is

XX + V + II

.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not

IIII

. Instead, the number four is written as

IV

. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as

IX

. There are six instances where subtraction is used:

I

can be placed before

V

(5) and

X

(10) to make 4 and 9.

X

can be placed before

L

(50) and

C

(100) to make 40 and 90.

C

can be placed before

D

(500) and

M

(1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

Example 1:

Input:

s = "III"

Output:

3

Explanation:

III = 3.

Example 2:

Input:

s = "LVIII"

Output:

58

Explanation:

L = 50, V= 5, III = 3.

Example 3:

Input:

s = "MCMXCIV"

Output:

1994

Explanation:

M = 1000, CM = 900, XC = 90 and IV = 4.

Constraints:

1 <= s.length <= 15

s

contains only the characters

('I', 'V', 'X', 'L', 'C', 'D', 'M')

.

It is

guaranteed

that

s

is a valid roman numeral in the range

[1, 3999]

.

Code Snippets

C++:

```
class Solution {
public:
int romanToInt(string s) {

}
};
```

Java:

```
class Solution {
public int romanToInt(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function romanToInt(s: string): number {

};
```

C#:

```
public class Solution {
public int RomanToInt(string s) {

}
}
```

C:

```
int romanToInt(char* s) {

}
```

Go:

```
func romanToInt(s string) int {

}
```

Kotlin:

```
class Solution {
fun romanToInt(s: String): Int {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn roman_to_int(s: String) -> i32 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int romanToInt(String s) {

}
```

```
}
```

Scala:

```
object Solution {
def romanToInt(s: String): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec roman_to_int(S :: unicode:unicode_binary()) -> integer().
roman_to_int(S) ->
.
```

Racket:

```
(define/contract (roman-to-int s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Roman to Integer
* Difficulty: Easy
* Tags: string, math, hash
*
```

```
* 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 romanToInt(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Roman to Integer
* Difficulty: Easy
* Tags: string, math, hash
*
* 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 romanToInt(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Roman to Integer
Difficulty: Easy
Tags: string, math, hash

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Roman to Integer
* Difficulty: Easy
* Tags: string, math, hash
*
* 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
* @return {number}
*/
var romanToInt = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Roman to Integer
* Difficulty: Easy
* Tags: string, math, hash
```

```
*
* 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 romanToInt(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Roman to Integer
* Difficulty: Easy
* Tags: string, math, hash
*
* 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 RomanToInt(string s) {

}
}
```

C Solution:

```
/*
* Problem: Roman to Integer
* Difficulty: Easy
* Tags: string, math, hash
*
* 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 romanToInt(char* s) {
```

```
}
```

Go Solution:

```
// Problem: Roman to Integer
// Difficulty: Easy
// Tags: string, math, hash
//
// 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 romanToInt(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun romanToInt(s: String): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Roman to Integer
// Difficulty: Easy
// Tags: string, math, hash
//
// 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 roman_to_int(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int romanToInt(String s) {

}
}
```

Scala Solution:

```
object Solution {
def romanToInt(s: String): Int = {
```

```
}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec roman_to_int(S :: unicode:unicode_binary()) -> integer().
roman_to_int(S) ->
.
```

Racket Solution:

```
(define/contract (roman-to-int s)
(-> string? exact-integer?)
)
```