Problem 8: String to Integer (atoi)

Problem Information

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

Problem Description

Implement the

myAtoi(string s)

function, which converts a string to a 32-bit signed integer.

The algorithm for

myAtoi(string s)

is as follows:

Whitespace

: Ignore any leading whitespace (

" "

).

Signedness

: Determine the sign by checking if the next character is

'-'

or

'+'

, assuming positivity if neither present.

Conversion

: Read the integer by skipping leading zeros until a non-digit character is encountered or the end of the string is reached. If no digits were read, then the result is 0.

Rounding

: If the integer is out of the 32-bit signed integer range

[-2

31

, 2

31

- 1]

, then round the integer to remain in the range. Specifically, integers less than

-2

31

should be rounded to

-2

31

, and integers greater than

2

31

- 1

should be rounded to

2

31

- 1

.

Return the integer as the final result.

Example 1:

Input:

s = "42"

Output:

42

Explanation:

The underlined characters are what is read in and the caret is the current reader position. Step 1: "42" (no characters read because there is no leading whitespace) ^ Step 2: "42" (no characters read because there is neither a '-' nor '+') ^ Step 3: "

42

" ("42" is read in) ^

Example 2:

Input:

s = " -042"

Output:

-42

Explanation:

Step 1: "

-042" (leading whitespace is read and ignored) ^ Step 2: "

-

042" ('-' is read, so the result should be negative) ^ Step 3: " -

042

" ("042" is read in, leading zeros ignored in the result) ^

Example 3:

Input:

s = "1337c0d3"

Output:

1337

Explanation:

Step 1: "1337c0d3" (no characters read because there is no leading whitespace) ^ Step 2: "1337c0d3" (no characters read because there is neither a '-' nor '+') ^ Step 3: "

1337

c0d3" ("1337" is read in; reading stops because the next character is a non-digit) ^

Example 4:

Input:

s = "0-1"

Output:

0

Explanation:

Step 1: "0-1" (no characters read because there is no leading whitespace) ^ Step 2: "0-1" (no characters read because there is neither a '-' nor '+') ^ Step 3: "

0

-1" ("0" is read in; reading stops because the next character is a non-digit) ^

Example 5:

Input:

s = "words and 987"

Output:

0

Explanation:

Reading stops at the first non-digit character 'w'.

Constraints:

0 <= s.length <= 200

s

consists of English letters (lower-case and upper-case), digits (

0-9

),

' '

,

'+'

,

'-'

, and

'.'

.

Code Snippets

C++:

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

}
};
```

Java:

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

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int myAtoi(char* s) {

}
```

Go:

```
func myAtoi(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (my-atoi s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: String to Integer (atoi)
* Difficulty: Medium
* 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:
int myAtoi(string s) {

}
};
```

Java Solution:

```
/**
* Problem: String to Integer (atoi)
* Difficulty: Medium
* 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 int myAtoi(String s) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: String to Integer (atoi)
Difficulty: Medium
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 myAtoi(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: String to Integer (atoi)
* Difficulty: Medium
* 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 {number}
*/
var myAtoi = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: String to Integer (atoi)
* Difficulty: Medium
* 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 myAtoi(s: string): number {

};
```

C# Solution:

```
/*
* Problem: String to Integer (atoi)
* Difficulty: Medium
* 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 int MyAtoi(string s) {

}
```

```
}
```

C Solution:

```
/*
* Problem: String to Integer (atoi)
* Difficulty: Medium
* 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
*/

int myAtoi(char* s) {

}
```

Go Solution:

```
// Problem: String to Integer (atoi)
// Difficulty: Medium
// 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 myAtoi(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: String to Integer (atoi)
// Difficulty: Medium
// 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 my_atoi(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (my-atoi s)
(-> string? exact-integer?)
)
```