Problem 65: Valid Number

Problem Information

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

Problem Description

Given a string

s

, return whether

s

is a

valid number

.

For example, all the following are valid numbers:

"2", "0089", "-0.1", "+3.14", "4.", "-.9", "2e10", "-90E3", "3e+7", "+6e-1", "53.5e93", "-123.456e789"

, while the following are not valid numbers:

"abc", "1a", "1e", "e3", "99e2.5", "--6", "-+3", "95a54e53"

.

Formally, a

valid number

is defined using one of the following definitions:

An

integer number

followed by an

optional exponent

.

A

decimal number

followed by an

optional exponent

.

An

integer number

is defined with an

optional sign

'-'

or

'+'

followed by

digits

.

A

decimal number

is defined with an

optional sign

'-'

or

'+'

followed by one of the following definitions:

Digits

followed by a

dot

'.'

.

Digits

followed by a

dot

'.'

followed by

digits

.

A

dot

'.'

followed by

digits

.

An

exponent

is defined with an

exponent notation

'e'

or

'E'

followed by an

integer number

.

The

digits

are defined as one or more digits.

Example 1:

Input:

s = "0"

Output:

true

Example 2:

Input:

s = "e"

Output:

false

Example 3:

Input:

s = "."

Output:

false

Constraints:

1 <= s.length <= 20

s

consists of only English letters (both uppercase and lowercase), digits (

0-9

), plus

'+'

, minus

'-'

, or dot

'.'

.

Code Snippets

C++:

```
class Solution {
public:
bool isNumber(string s) {

}
};
```

Java:

```
class Solution {
public boolean isNumber(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function isNumber(s: string): boolean {

};
```

C#:

```
public class Solution {
public bool IsNumber(string s) {

}
}
```

C:

```
bool isNumber(char* s) {

}
```

Go:

```
func isNumber(s string) bool {

}
```

Kotlin:

```
class Solution {
fun isNumber(s: String): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
bool isNumber(String s) {

}
}
```

Scala:

```
object Solution {
def isNumber(s: String): Boolean = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec is_number(S :: unicode:unicode_binary()) -> boolean().
is_number(S) ->
.
```

Racket:

```
(define/contract (is-number s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Valid Number
* Difficulty: Hard
* 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:
bool isNumber(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Valid Number
* Difficulty: Hard
* 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 boolean isNumber(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Valid Number
Difficulty: Hard
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 isNumber(self, s: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Valid Number
* Difficulty: Hard
* 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 {boolean}
*/
var isNumber = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Valid Number
* Difficulty: Hard
* 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 isNumber(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Valid Number
* Difficulty: Hard
* 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 bool IsNumber(string s) {

}
}
```

C Solution:

```
/*
* Problem: Valid Number
* Difficulty: Hard
* 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
*/

bool isNumber(char* s) {

}
```

Go Solution:

```
// Problem: Valid Number
// Difficulty: Hard
// 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 isNumber(s string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isNumber(s: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isNumber(_ s: String) -> Bool {
}
}
```

Rust Solution:

```
// Problem: Valid Number
// Difficulty: Hard
// 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 is_number(s: String) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
bool isNumber(String s) {

}
}
```

Scala Solution:

```
object Solution {
def isNumber(s: String): Boolean = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec is_number(S :: unicode:unicode_binary()) -> boolean().
is_number(S) ->
.
```

Racket Solution:

```
(define/contract (is-number s)
(-> string? boolean?)
)
```