Problem 468: Validate IP Address

Problem Information

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

Problem Description

Given a string

queryIP

, return

"IPv4"

if IP is a valid IPv4 address,

"IPv6"

if IP is a valid IPv6 address or

"Neither"

if IP is not a correct IP of any type.

A valid IPv4

address is an IP in the form

"x

1

.x

2

.x

3

.x

4

"

where

0 <= x

i

<= 255

and

x

i

cannot contain

leading zeros. For example,

"192.168.1.1"

and

"192.168.1.0"

are valid IPv4 addresses while

"192.168.01.1"

,

"192.168.1.00"

, and

"192.168@1.1"

are invalid IPv4 addresses.

A valid IPv6

address is an IP in the form

"x

1

:x

2

:x

3

:x

4

:x

5

:x

6

:x

7

:x

8

"

where:

1 <= x

i

.length <= 4

x

i

is a

hexadecimal string

which may contain digits, lowercase English letter (

'a'

to

'f'

) and upper-case English letters (

'A'

to

'F'

).

Leading zeros are allowed in

x

i

.

For example, "

2001:0db8:85a3:0000:0000:8a2e:0370:7334"

and "

2001:db8:85a3:0:0:8A2E:0370:7334"

are valid IPv6 addresses, while "

2001:0db8:85a3::8A2E:037j:7334"

and "

02001:0db8:85a3:0000:0000:8a2e:0370:7334"

are invalid IPv6 addresses.

Example 1:

Input:

queryIP = "172.16.254.1"

Output:

"IPv4"

Explanation:

This is a valid IPv4 address, return "IPv4".

Example 2:

Input:

queryIP = "2001:0db8:85a3:0:0:8A2E:0370:7334"

Output:

"IPv6"

Explanation:

This is a valid IPv6 address, return "IPv6".

Example 3:

Input:

queryIP = "256.256.256.256"

Output:

"Neither"

Explanation:

This is neither a IPv4 address nor a IPv6 address.

Constraints:

queryIP

consists only of English letters, digits and the characters

'.'

and

':'

.

Code Snippets

C++:

```
class Solution {
public:
string validIPAddress(string queryIP) {

}
};
```

Java:

```
class Solution {
public String validIPAddress(String queryIP) {

}
}
```

Python3:

```
class Solution:
def validIPAddress(self, queryIP: str) -> str:
```

Python:

```
class Solution(object):
def validIPAddress(self, queryIP):
"""
:type queryIP: str
```

```
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} queryIP
* @return {string}
*/
var validIPAddress = function(queryIP) {

};
```

TypeScript:

```
function validIPAddress(queryIP: string): string {

};
```

C#:

```
public class Solution {
public string ValidIPAddress(string queryIP) {

}
}
```

C:

```
char* validIPAddress(char* queryIP) {

}
```

Go:

```
func validIPAddress(queryIP string) string {

}
```

Kotlin:

```
class Solution {
fun validIPAddress(queryIP: String): String {

}
}
```

Swift:

```
class Solution {
func validIPAddress(_ queryIP: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn valid_ip_address(query_ip: String) -> String {

}
}
```

Ruby:

```
# @param {String} query_ip
# @return {String}
def valid_ip_address(query_ip)

end
```

PHP:

```
class Solution {

/**
* @param String $queryIP
* @return String
*/
function validIPAddress($queryIP) {

}
}
```

Dart:

```
class Solution {
String validIPAddress(String queryIP) {

}
}
```

Scala:

```
object Solution {
def validIPAddress(queryIP: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec valid_ip_address(query_ip :: String.t) :: String.t
def valid_ip_address(query_ip) do

end
end
```

Erlang:

```
-spec valid_ip_address(QueryIP :: unicode:unicode_binary()) ->
unicode:unicode_binary().
valid_ip_address(QueryIP) ->
.
```

Racket:

```
(define/contract (valid-ip-address queryIP)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Validate IP Address
* 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:
string validIPAddress(string queryIP) {

}
};
```

Java Solution:

```
/**
* Problem: Validate IP Address
* 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 String validIPAddress(String queryIP) {

}
}
```

Python3 Solution:

```
"""
Problem: Validate IP Address
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 validIPAddress(self, queryIP: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def validIPAddress(self, queryIP):
"""
:type queryIP: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Validate IP Address
* 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} queryIP
* @return {string}
*/
var validIPAddress = function(queryIP) {

};
```

TypeScript Solution:

```
/**
* Problem: Validate IP Address
* 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 validIPAddress(queryIP: string): string {

};
```

C# Solution:

```
/*
* Problem: Validate IP Address
* 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 string ValidIPAddress(string queryIP) {

}
}
```

C Solution:

```
/*
* Problem: Validate IP Address
* 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
```

```
*/

char* validIPAddress(char* queryIP) {

}
```

Go Solution:

```
// Problem: Validate IP Address
// 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 validIPAddress(queryIP string) string {

}
```

Kotlin Solution:

```
class Solution {
fun validIPAddress(queryIP: String): String {

}
}
```

Swift Solution:

```
class Solution {
func validIPAddress(_ queryIP: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Validate IP Address
// 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 valid_ip_address(query_ip: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} query_ip
# @return {String}
def valid_ip_address(query_ip)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $queryIP
* @return String
*/
function validIPAddress($queryIP) {

}
}
```

Dart Solution:

```
class Solution {
String validIPAddress(String queryIP) {

}
}
```

Scala Solution:

```
object Solution {
def validIPAddress(queryIP: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec valid_ip_address(query_ip :: String.t) :: String.t
def valid_ip_address(query_ip) do

end
end
```

Erlang Solution:

```
-spec valid_ip_address(QueryIP :: unicode:unicode_binary()) ->
unicode:unicode_binary().
valid_ip_address(QueryIP) ->
.
```

Racket Solution:

```
(define/contract (valid-ip-address queryIP)
(-> string? string?)
)
```