Problem 1108: Defanging an IP Address

Problem Information

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

Problem Description

Given a valid (IPv4) IP

address

, return a defanged version of that IP address.

A

defanged IP address

replaces every period

"."

with

"[.]"

.

Example 1:

Input:

address = "1.1.1.1"

Output:

"1[.]1[.]1[.]1"

Example 2:

Input:

address = "255.100.50.0"

Output:

"255[.]100[.]50[.]0"

Constraints:

The given

address

is a valid IPv4 address.

Code Snippets

C++:

```
class Solution {
public:
string defangIPaddr(string address) {

}
};
```

Java:

```
class Solution {
public String defangIPaddr(String address) {

}
}
```

Python3:

```
class Solution:
def defangIPaddr(self, address: str) -> str:
```

Python:

```
class Solution(object):
def defangIPaddr(self, address):
"""
:type address: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} address
* @return {string}
*/
var defangIPaddr = function(address) {

};
```

TypeScript:

```
function defangIPaddr(address: string): string {

};
```

C#:

```
public class Solution {
public string DefangIPaddr(string address) {

}
}
```

C:

```


char * defangIPaddr(char * address){

}
```

Go:

```
func defangIPaddr(address string) string {

}
```

Kotlin:

```
class Solution {
fun defangIPaddr(address: String): String {

}
}
```

Swift:

```
class Solution {
func defangIPaddr(_ address: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn defang_i_paddr(address: String) -> String {

}
}
```

Ruby:

```
# @param {String} address
# @return {String}
def defang_i_paddr(address)

end
```

PHP:

```
class Solution {

/**
* @param String $address
* @return String
*/
function defangIPaddr($address) {


}
}
```

Scala:

```
object Solution {
def defangIPaddr(address: String): String = {


}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Defanging an IP Address
* Difficulty: Easy
* 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 defangIPaddr(string address) {


}
};
```

Java Solution:

```
/**
* Problem: Defanging an IP Address
* Difficulty: Easy
* 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 defangIPaddr(String address) {

}
}
```

Python3 Solution:

```
"""
Problem: Defanging an IP Address
Difficulty: Easy
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 defangIPaddr(self, address: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def defangIPaddr(self, address):
"""
:type address: str
:rtype: str
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Defanging an IP Address
* Difficulty: Easy
* 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} address
* @return {string}
*/
var defangIPaddr = function(address) {

};
```

TypeScript Solution:

```
/**
* Problem: Defanging an IP Address
* Difficulty: Easy
* 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 defangIPaddr(address: string): string {

};
```

C# Solution:

```
/*
* Problem: Defanging an IP Address
* Difficulty: Easy
* 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 DefangIPaddr(string address) {

}
}
```

C Solution:

```
/*
* Problem: Defanging an IP Address
* Difficulty: Easy
* 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 * defangIPaddr(char * address){

}
```

Go Solution:

```
// Problem: Defanging an IP Address
// Difficulty: Easy
// 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 defangIPaddr(address string) string {

}
```

Kotlin Solution:

```
class Solution {
fun defangIPaddr(address: String): String {

}
}
```

Swift Solution:

```
class Solution {
func defangIPaddr(_ address: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Defanging an IP Address
// Difficulty: Easy
// 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 defang_i_paddr(address: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} address
# @return {String}
def defang_i_paddr(address)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $address
* @return String
*/
function defangIPaddr($address) {

}
}
```

Scala Solution:

```
object Solution {
def defangIPaddr(address: String): String = {

}
}
```