Problem 93: Restore IP Addresses

Problem Information

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

Problem Description

A

valid IP address

consists of exactly four integers separated by single dots. Each integer is between

0

and

255

(

inclusive

) and cannot have leading zeros.

For example,

"0.1.2.201"

and

"192.168.1.1"

are

valid

IP addresses, but

"0.011.255.245"

,

"192.168.1.312"

and

"192.168@1.1"

are

invalid

IP addresses.

Given a string

s

containing only digits, return

all possible valid IP addresses that can be formed by inserting dots into

s

. You are

not

allowed to reorder or remove any digits in

s

. You may return the valid IP addresses in

any

order.

Example 1:

Input:

s = "25525511135"

Output:

["255.255.11.135","255.255.111.35"]

Example 2:

Input:

s = "0000"

Output:

["0.0.0.0"]

Example 3:

Input:

s = "101023"

Output:

["1.0.10.23","1.0.102.3","10.1.0.23","10.10.2.3","101.0.2.3"]

Constraints:

1 <= s.length <= 20

s

consists of digits only.

Code Snippets

C++:

```
class Solution {
public:
vector<string> restoreIpAddresses(string s) {

}
};
```

Java:

```
class Solution {
public List<String> restoreIpAddresses(String s) {

}
}
```

Python3:

```
class Solution:
def restoreIpAddresses(self, s: str) -> List[str]:
```

Python:

```
class Solution(object):
def restoreIpAddresses(self, s):
"""
:type s: str
:rtype: List[str]
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string[]}
*/
var restoreIpAddresses = function(s) {

};
```

TypeScript:

```
function restoreIpAddresses(s: string): string[] {

};
```

C#:

```
public class Solution {
public IList<string> RestoreIpAddresses(string s) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** restoreIpAddresses(char* s, int* returnSize) {

}
```

Go:

```
func restoreIpAddresses(s string) []string {

}
```

Kotlin:

```
class Solution {
fun restoreIpAddresses(s: String): List<String> {

}
}
```

Swift:

```
class Solution {
func restoreIpAddresses(_ s: String) -> [String] {

}
}
```

Rust:

```
impl Solution {
pub fn restore_ip_addresses(s: String) -> Vec<String> {

}
}
```

Ruby:

```
# @param {String} s
# @return {String[]}
def restore_ip_addresses(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String[]
*/
function restoreIpAddresses($s) {

}
}
```

Dart:

```
class Solution {
List<String> restoreIpAddresses(String s) {

}
}
```

Scala:

```
object Solution {
def restoreIpAddresses(s: String): List[String] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec restore_ip_addresses(s :: String.t) :: [String.t]
def restore_ip_addresses(s) do

end
end
```

Erlang:

```
-spec restore_ip_addresses(S :: unicode:unicode_binary()) ->
[unicode:unicode_binary()].
restore_ip_addresses(S) ->
.
```

Racket:

```
(define/contract (restore-ip-addresses s)
(-> string? (listof string?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Restore IP Addresses
* 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:
vector<string> restoreIpAddresses(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Restore IP Addresses
* 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 List<String> restoreIpAddresses(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Restore IP Addresses
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 restoreIpAddresses(self, s: str) -> List[str]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def restoreIpAddresses(self, s):
"""
:type s: str
:rtype: List[str]
"""
```

JavaScript Solution:

```
/**
* Problem: Restore IP Addresses
* 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 {string[]}
*/
var restoreIpAddresses = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Restore IP Addresses
* 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 restoreIpAddresses(s: string): string[] {

};
```

C# Solution:

```
/*
* Problem: Restore IP Addresses
* 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 IList<string> RestoreIpAddresses(string s) {

}
}
```

C Solution:

```
/*
* Problem: Restore IP Addresses
* 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
```

```
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** restoreIpAddresses(char* s, int* returnSize) {

}
```

Go Solution:

```
// Problem: Restore IP Addresses
// 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 restoreIpAddresses(s string) []string {

}
```

Kotlin Solution:

```
class Solution {
fun restoreIpAddresses(s: String): List<String> {

}
}
```

Swift Solution:

```
class Solution {
func restoreIpAddresses(_ s: String) -> [String] {

}
}
```

Rust Solution:

```
// Problem: Restore IP Addresses
// 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 restore_ip_addresses(s: String) -> Vec<String> {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String[]}
def restore_ip_addresses(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String[]
*/
function restoreIpAddresses($s) {

}
}
```

Dart Solution:

```
class Solution {
List<String> restoreIpAddresses(String s) {

}
}
```

Scala Solution:

```
object Solution {
def restoreIpAddresses(s: String): List[String] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec restore_ip_addresses(s :: String.t) :: [String.t]
def restore_ip_addresses(s) do

end
end
```

Erlang Solution:

```
-spec restore_ip_addresses(S :: unicode:unicode_binary()) ->
[unicode:unicode_binary()].
restore_ip_addresses(S) ->
.
```

Racket Solution:

```
(define/contract (restore-ip-addresses s)
(-> string? (listof string?))
)
```