

## Validate IP Address

In this problem, your job to write a function to check whether a input string is a valid IPv4 address or IPv6 address or neither.

**IPv4** addresses are canonically represented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots ("."), e.g., `172.16.254.1`;

Besides, you need to keep in mind that leading zeros in the IPv4 is illegal. For example, the address `172.16.254.01` is illegal.

**IPv6** addresses are represented as eight groups of four hexadecimal digits, each group representing 16 bits. The groups are separated by colons (":"). For example, the address `2001:0db8:85a3:0000:0000:8a2e:0370:7334` is a legal one. Also, we could omit some leading zeros among four hexadecimal digits and some low-case characters in the address to upper-case ones, so `2001:db8:85a3:0:0:8A2E:0370:7334` is also a valid IPv6 address(Omit leading zeros and using upper cases).

However, we don't replace a consecutive group of zero value with a single empty group using two consecutive colons (":") to pursue simplicity. For example, `2001:0db8:85a3::8A2E:0370:7334` is an invalid IPv6 address.

Besides, you need to keep in mind that extra leading zeros in the IPv6 is also illegal. For example, the address `02001:0db8:85a3:0000:0000:8a2e:0370:7334` is also illegal.

**Note:** You could assume there is no extra space in the test cases and there may some special characters in the input string.

### Example 1:

**Input:** "172.16.254.1"

**Output:** "IPv4"

**Explanation:** This is a valid IPv4 address, return "IPv4".

### Example 2:

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

**Output:** "IPv6"

**Explanation:** This is a valid IPv6 address, return "IPv6".

### Example 3:

**Input:** "256.256.256.256"

**Output:** "Neither"

**Explanation:** This is neither a IPv4 address nor a IPv6 address.

## Solution 1

```
public String validIPAddress(String IP) {
    if(isValidIPv4(IP)) return "IPv4";
    else if(isValidIPv6(IP)) return "IPv6";
    else return "Neither";
}

public boolean isValidIPv4(String ip) {
    if(ip.length()<7) return false;
    if(ip.charAt(0)=='.') return false;
    if(ip.charAt(ip.length()-1)=='.') return false;
    String[] tokens = ip.split("\\.");
    if(tokens.length!=4) return false;
    for(String token:tokens) {
        if(!isValidIPv4Token(token)) return false;
    }
    return true;
}

public boolean isValidIPv4Token(String token) {
    if(token.startsWith("0") && token.length()>1) return false;
    try {
        int parsedInt = Integer.parseInt(token);
        if(parsedInt<0 || parsedInt>255) return false;
        if(parsedInt==0 && token.charAt(0)!='0') return false;
    } catch(NumberFormatException nfe) {
        return false;
    }
    return true;
}

public boolean isValidIPv6(String ip) {
    if(ip.length()<15) return false;
    if(ip.charAt(0)==':') return false;
    if(ip.charAt(ip.length()-1)==':') return false;
    String[] tokens = ip.split(":");
    if(tokens.length!=8) return false;
    for(String token: tokens) {
        if(!isValidIPv6Token(token)) return false;
    }
    return true;
}

public boolean isValidIPv6Token(String token) {
    if(token.length()>4 || token.length()==0) return false;
    char[] chars = token.toCharArray();
    for(char c:chars) {
        boolean isDigit = c>=48 && c<=57;
        boolean isUppercaseAF = c>=65 && c<=70;
        boolean isLowercaseAF = c>=97 && c<=102;
        if(!(isDigit || isUppercaseAF || isLowercaseAF))
            return false;
    }
    return true;
}
```

written by [fabrizio3](#) original link [here](#)

## Solution 2

```
const ip4 = /^[1-9]\d{0,2}|0(?:\.[1-9]\d{0,2}|0){3}$/;
const ip6 = /^[0-9a-fA-F]{1,4}(\:[0-9a-fA-F]{1,4}){7}$/;

var validIPAddress = function(IP) {
  const isIp4 = ip4.exec(IP);
  if (isIp4 && isIp4.slice(1).every(d => parseInt(d, 10) < 256))
    return 'IPv4';

  const isIp6 = ip6.exec(IP);
  if (isIp6)
    return 'IPv6';

  return 'Neither';
};
```

written by [dettier](#) original link [here](#)

## Solution 3

```
class Solution(object):
    def validIPAddress(self, IP):
        def is_hex(s):
            hex_digits = set("0123456789abcdefABCDEF")
            for char in s:
                if not (char in hex_digits):
                    return False
            return True
        ary = IP.split('.')
        if len(ary) == 4:
            for i in xrange(len(ary)):
                if not ary[i].isdigit() or not 0 <= int(ary[i]) < 256 or (ary[i][0] == '0' and len(ary[i]) > 1):
                    return "Neither"
            return "IPv4"
        ary = IP.split(':')
        if len(ary) == 8:
            for i in xrange(len(ary)):
                tmp = ary[i]
                if len(tmp) == 0 or not len(tmp) <= 4 or not is_hex(tmp):
                    return "Neither"
            return "IPv6"
        return "Neither"
```

written by [Ipeq1](#) original link [here](#)

From [LeetCoder](#).