Generate IP Addresses

Sunday, December 12, 2021 12:38 PM

Generate IP Addresses

Medium Accuracy: 43.42% Submissions: 13596 Points: 4

Given a string **S** containing only digits, Your task is to complete the function **genIp()** which returns a vector containing all possible combination of valid IPv4 ip address and takes only a string **S** as its only argument.

Note : Order doesn't matter.

For string 11211 the ip address possible are

1.1.2.11

1.1.21.1

1.12.1.1

11.2.1.1

Example 1:

```
Input:
S = 1111
Output: 1.1.1.1
```

Your Task:

Your task is to complete the function **genIp()** which returns a vector containing all possible combination of valid IPv4 ip address in sorted order and takes only a string **S** as its only argument.

Expected Time Complexity: O(N * N * N * N)

Expected Auxiliary Space: O(N * N * N * N)

Constraints:

1<=N<=16

here, N = length of S.

S only contains digits(i.e. 0-9)

From <https://practice.geeksforgeeks.org/problems/generate-ip-addresses/1>

Company Tags
Amazon

From <https://practice.geeksforgeeks.org/problems/generate-ip-addresses/1>

Given a string containing only digits, restore it by returning all possible valid IP address combinations.

A valid IP address must be in the form of A.B.C.D, where A, B, C, and D are numbers from 0-255. The numbers cannot be 0 prefixed unless they are 0.

Examples :

```
Input: 25525511135
Output: [“255.255.11.135”, “255.255.111.35”]
Explanation:
These are the only valid possible
IP addresses.
Input: "25505011535"
Output: []
Explanation:
We cannot generate a valid IP
address with this string.
```

First, we will place 3 dots in the given string and then try out all the possible combinations for the 3 dots.

Corner case for validity:

```
For string "25011255255"
25.011.255.255 is not valid as 011 is not valid.
25.11.255.255 is not valid either as you are not
allowed to change the string.
250.11.255.255 is valid.
```

Recommended: Please try your approach on ***{IDE}*** first, before moving on to the solution.

Approach: Split the string with ‘ . ‘ and then check for all corner cases. Before entering the loop, check the size of the string. Generate all the possible combinations using looping through the string. If IP is found to be valid then return the IP address, else simply return the empty list.

Below is the implementation of the above approach:

- C++
- Java
- Python3

```
// Java Program to generate all possible
// valid IP addresses from given string
import java.util.*;

class GFG {

    // Function to restore Ip Addresses
    public static ArrayList<String>
    restoreIpAddresses(String A)
    {
        if (A.length() < 3 || A.length() > 12)
            return new ArrayList<>();
        return convert(A);
    }

    private static ArrayList<String>
    convert(String s)
    {
        ArrayList<String> l = new ArrayList<>();
        int size = s.length();

        String snew = s;

        for (int i = 1; i < size - 2;
             i++) {
            for (int j = i + 1;
                 j < size - 1; j++) {
                for (int k = j + 1;
                     k < size; k++) {
                    snew
                        = snew.substring(0, k) + "."
                          + snew.substring(k);
                    snew
                        = snew.substring(0, j) + "."
                          + snew.substring(j);
                    snew
                        = snew.substring(0, i) + "."
                          + snew.substring(i);

                    if (isValid(snew)) {
                        l.add(snew);
                    }
                    snew = s;
                }
            }
        }

        Collections.sort(l, new Comparator<String>() {
            public int compare(String o1, String o2)
            {
                String a1[] = o1.split("[.]");
                String a2[] = o2.split("[.]");

                int result = -1;
                for (int i = 0; i < 4
                                && result != 0;
                     i++) {
                    result = a1[i].compareTo(a2[i]);
                }
                return result;
            }
        });
        return l;
    }

    private static boolean isValid(String ip)
    {
        String a[] = ip.split("[.]");
        for (String s : a) {
            int i = Integer.parseInt(s);
            if (s.length() > 3 || i < 0 || i > 255) {
                return false;
            }
            if (s.length() > 1 && i == 0)
                return false;
            if (s.length() > 1 && i != 0
                && s.charAt(0) == '0')
                return false;
        }

        return true;
    }

    // Driver Code
    public static void main(String[] args)
    {
        System.out.println(
            restoreIpAddresses(
                "25525511135")
                .toString());
    }
}

// This code is contributed by Nidhi Hooda.
```

From <https://www.geeksforgeeks.org/program-generate-possible-valid-ip-addresses-given-string/>

Output

```
255.255.11.135
255.255.111.35
```

Complexity Analysis:

- **Time Complexity:** O(n^3), where n is the length of the string
  Three nested traversal of the string is needed, where n is always less than 12.
- **Auxiliary Space:** O(n).
  As as extra space is needed.
  Another Efficient Approach (Dynamic Programming): There is a dp approach exist for this problem and can be solved in time complexity O(n*4*3)=O(12n)=O(n) and space complexity O(4n).

  Approach: We know that there are only 4 parts of the IP. We start iterating from the end of string and goes to the start of string. We create a dp 2D-array of size (4 x N). There can be only 2 values in the dp array i.e. 1(true) or 0(false). dp[0][i] tells if we can create 1 part of IP from the substring starting from i to end of string. Similarly, dp[1][i] tells if we can create 2 parts of IP from the substring starting from i to end of string.

  After creating the dp array, we start creating the valid IP addresses. We start from the bottom left corner of the 2D dp array. We only iterate 12 times(worst case) but those also will be the valid IP addresses because we only form valid IP addresses.

- Java

```java
// Java Program to generate all possible
// valid IP addresses from given string
import java.util.*;

class GFG
{
    public static ArrayList<String> list;

    // Function to restore Ip Addresses
    public static ArrayList<String>
    restoreIpAddresses(String s)
    {
        int n = s.length();
        list = new ArrayList<>();
        if (n < 4 || n > 12)
            return list;

         // initialize the dp array
        int dp[][] = new int[4][n];
        for (int i = 0; i < 4; i++)
        {
            for (int j = n - 1; j >= 0; j--)
            {
                if (i == 0)
                {
```

```
                // take the substring
                String sub = s.substring(j);
                if (isValid(sub))
                {
                    dp[i][j] = 1;
                }
                else if (j < n - 3)
                {
                    break;
                }
            }
            else
            {
                if (j <= n - i)
                {
                    for (int k = 1;
                         k <= 3 && j + k <= n;
                         k++)
                    {
                        String temp
                            = s.substring(j, j + k);
                        if (isValid(temp))
                        {
                            if (j + k < n
                                && dp[i - 1][j + k]
                                       == 1)
                            {
                                dp[i][j] = 1;
                                break;
                            }
                        }
                    }
                }
            }
        }
    }

    if (dp[3][0] == 0)
        return list;


    // Call function createfromDp
    createIpFromDp(dp, 3, 0, s, "");
    return list;
}

public static void createIpFromDp(int dp[][],
                                  int r,
                                  int c, String s,
                                  String ans)
{
    if (r == 0)
    {
        ans += s.substring(c);
        list.add(ans);
```

```
            return;
        }
        for (int i = 1;
             i <= 3 && c + i < s.length();
             i++)
        {
            if (isValid(s.substring(c, c + i))
                && dp[r - 1] == 1)
            {
                createIpFromDp(dp, r - 1, c + i, s,
                               ans +
                               s.substring(c, c + i)
                               + ".");
            }
        }
    }


    private static boolean isValid(String ip)
    {
        String a[] = ip.split("[.]");
        for (String s : a)
        {
            int i = Integer.parseInt(s);
            if (s.length() > 3 || i < 0 || i > 255)
            {
                return false;
            }
            if (s.length() > 1 && i == 0)
                return false;
            if (s.length() > 1 && i != 0
                && s.charAt(0) == '0')
                return false;
        }

        return true;
    }

    // Driver Code
    public static void main(String[] args)
    {
        // Function call
        System.out.println(
            restoreIpAddresses("25525511135").toString());
    }
}

// This code is contributed by Nidhi Hooda.
```
Output
```
[255.255.11.135, 255.255.111.35]
```
Complexity Analysis:

- **Time Complexity**: O(n), where n is the length of the string. The dp array creation would take O(4*n*3) = O(12n) = O(n). Valid IP creation from dp array would take O(n).
- **Auxiliary Space**: O(n). As dp array has extra space of size (4 x n). It means O(4n).

Another Approach: **(Using Recursion)**

- C++

```
#include <iostream>
#include <vector>
using namespace std;

void solve(string s, int i, int j, int level, string temp,
           vector<string>& res)
{
    if (i == (j + 1) && level == 5) {
        res.push_back(temp.substr(1));
    }

    // Digits of a number ranging 0-255 can lie only between
    // 0-3
    for (int k = i; k < i + 3 && k <= j; k++) {
        string ad = s.substr(i, k - i + 1);

        // Return if sting starting with '0' or it is
        // greater than 255.
        if (s[i] == '0' || stol(ad) > 255)
            return;

        // Recursively call for another level.
        solve(s, k + 1, j, level + 1, temp + '.' + ad, res);
    }
}

int main()
{
    string s = "25525511135";
    int n = s.length();

    vector<string> ans;

    solve(s, 0, n - 1, 1, "", ans);

    for (string s : ans)
        cout << s << endl;

    return 0;
}
```

Output

```
255.255.11.135
255.255.111.35
```

From <https://www.geeksforgeeks.org/program-generate-possible-valid-ip-addresses-given-string/>