```
1 class Solution {
2     public List<String> ipToCIDR(String ip, int n) {
3
4     }
5 }
```

751. IP to CIDR

Medium 9 30 Add to List Share

An **IP address** is a formatted 32-bit unsigned integer where each group of 8 bits is printed as a decimal number and the dot character `'.'` splits the groups.

- For example, the binary number `00001111 10001000 11111111 01101011` (spaces added for clarity) formatted as an IP address would be `"15.136.255.107"`.

A **CIDR block** is a format used to denote a specific set of IP addresses. It is a string consisting of a base IP address, followed by a slash, followed by a prefix length `k`. The addresses it covers are all the IPs whose **first** `k` **bits** are the same as the base IP address.

- For example, `"123.45.67.89/20"` is a CIDR block with a prefix length of `20`. Any IP address whose binary representation matches `01111011 00101101 0100xxxx xxxxxxxx`, where `x` can be either `0` or `1`, is in the set covered by the CIDR block.

You are given a start IP address `ip` and the number of IP addresses we need to cover `n`. Your goal is to use **as few CIDR blocks as possible** to cover all the IP addresses in the **inclusive** range `[ip, ip + n - 1]` **exactly**. No other IP addresses outside of the range should be covered.

Return *the* ***shortest*** *list of* ***CIDR blocks*** *that covers the range of IP addresses. If there are multiple answers, return* ***any*** *of them.*

Example 1:

```
Input: ip = "255.0.0.7", n = 10
Output: ["255.0.0.7/32","255.0.0.8/29","255.0.0.16/32"]
Explanation:
The IP addresses that need to be covered are:
- 255.0.0.7  -> 11111111 00000000 00000000 00000111
- 255.0.0.8  -> 11111111 00000000 00000000 00001000
- 255.0.0.9  -> 11111111 00000000 00000000 00001001
- 255.0.0.10 -> 11111111 00000000 00000000 00001010
- 255.0.0.11 -> 11111111 00000000 00000000 00001011
- 255.0.0.12 -> 11111111 00000000 00000000 00001100
- 255.0.0.13 -> 11111111 00000000 00000000 00001101
- 255.0.0.14 -> 11111111 00000000 00000000 00001110
- 255.0.0.15 -> 11111111 00000000 00000000 00001111
- 255.0.0.16 -> 11111111 00000000 00000000 00010000
The CIDR block "255.0.0.7/32" covers the first address.
The CIDR block "255.0.0.8/29" covers the middle 8 addresses (binary format of
11111111 00000000 00000000 00001xxx).
The CIDR block "255.0.0.16/32" covers the last address.
Note that while the CIDR block "255.0.0.0/28" does cover all the addresses, it
also includes addresses outside of the range, so we cannot use it.
```

Example 2:

```
Input: ip = "117.145.102.62", n = 8
Output: ["117.145.102.62/31","117.145.102.64/30","117.145.102.68/31"]
```

Constraints:

- `7 <= ip.length <= 15`
- `ip` is a valid **IPv4** on the form `"a.b.c.d"` where `a`, `b`, `c`, and `d` are integers in the range `[0, 255]`.
- `1 <= n <= 1000`
- Every implied address `ip + x` (for `x < n`) will be a valid IPv4 address.

Accepted 12,694 | Submissions 22,716

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Databricks | 2

Related Topics

String Bit Manipulation

Similar Questions

Restore IP Addresses Medium

Validate IP Address Medium

Hide Hint 1

Convert the ip addresses to and from (long) integers. You want to know what is the most addresses you can put in this block starting from the "start" ip, up to n. It is the smallest between the lowest bit of start and the highest bit of n. Then, repeat this process with a new start and n.