751. IP to CIDR

Dec. 23, 2017 | 11.7K views

Average Rating: 1.61 (28 votes)

Given a start IP address `ip` and a number of ips we need to cover `n`, return a representation of the range as a list (of smallest possible length) of CIDR blocks.

A CIDR block is a string consisting of an IP, followed by a slash, and then the prefix length. For example: "123.45.67.89/20". That prefix length "20" represents the number of common prefix bits in the specified range.

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 initial ip address, when converted to binary, looks like this (spaces added for cl
255.0.0.7 -> 11111111 00000000 00000000 00000111
The address "255.0.0.7/32" specifies all addresses with a common prefix of 32 bits to
ie. just this one address.

The address "255.0.0.8/29" specifies all addresses with a common prefix of 29 bits to
255.0.0.8 -> 11111111 00000000 00000000 00001000
Addresses with common prefix of 29 bits are:
11111111 00000000 00000000 00001000
11111111 00000000 00000000 00001001
11111111 00000000 00000000 00001010
11111111 00000000 00000000 00001011
11111111 00000000 00000000 00001100
11111111 00000000 00000000 00001101
11111111 00000000 00000000 00001110
11111111 00000000 00000000 00001111

The address "255.0.0.16/32" specifies all addresses with a common prefix of 32 bits to
ie. just 11111111 00000000 00000000 00010000.

In total, the answer specifies the range of 10 ips starting with the address 255.0.0.7

There were other representations, such as:
["255.0.0.7/32","255.0.0.8/30", "255.0.0.12/30", "255.0.0.16/32"],
but our answer was the shortest possible.

Also note that a representation beginning with say, "255.0.0.7/30" would be incorrect,
because it includes addresses like 255.0.0.4 = 11111111 00000000 00000000 00000100
that are outside the specified range.
```

Note:

1. `ip` will be a valid IPv4 address.
2. Every implied address `ip + x` (for `x < n`) will be a valid IPv4 address.
3. `n` will be an integer in the range `[1, 1000]`.

Approach #1: Direct [Accepted]

Intuition

This problem is about performing the steps directly as written. The tricky part is managing the bit manipulations involved.

Let's ask the question: for a number `n` of ip addresses desired, and the starting address `ip` of that range, what is the CIDR block representing the most ip addresses in that range starting at `ip`? Evidently, this greedy approach will work, and we can keep repeating this until we are done, so let's just focus on creating one largest block.

Algorithm

We'll need to be able to convert `ip` addresses back and forth to integers (`long`). We can do this with some basic manipulations - see the code for more details.

Then, with an ip address like `255.0.0.24` converted to `start`, it ends in the binary `00011000`. There are some cases. If `n >= 8`, then we should use the entire block `255.0.0.24/29`. Otherwise, we can only take a number of addresses equal to the largest power of 2 less than or equal to `n`.

In a more general setting, we use the bit lengths of both `n` and `start & -start` (the lowest bit of `start`) to compute the `mask` which represents $2^{32-\text{mask}}$ ip addresses. Then, we adjust `start` and `n` appropriately.

In Java and C++, we should be careful to use `long` data types to represent the converted ip addresses, since the number could exceed 2^{31}.

Java | Python — Copy

```python
class Solution(object):
    def ipToInt(self, ip):
        ans = 0
        for x in ip.split('.'):
            ans = 256 * ans + int(x)
        return ans

    def intToIP(self, x):
        return ".".join(str((x >> i) % 256)
                        for i in (24, 16, 8, 0))

    def ipToCIDR(self, ip, n):
        start = self.ipToInt(ip)
        ans = []
        while n:
            mask = max(33 - (start & -start).bit_length(),
                       33 - n.bit_length())
            ans.append(self.intToIP(start) + '/' + str(mask))
            start += 1 << (32 - mask)
            n -= 1 << (32 - mask)
        return ans
```

Complexity Analysis

- Time Complexity: $O(N)$ where N is the length of `nums`.
- Space Complexity: $O(1)$, the space used by our `int` variables.

Analysis written by: @awice.

Rate this article:

Previous Next

Comments: 10

Sort By

Type comment here... (Markdown is supported)

Preview Post

Joshua924 ★817 October 24, 2018 10:39 AM

This problem is just wield. It wants to test your understanding of bit manipulation, but to do that it creates way too complicated a context.

20 Share Reply

zsolt ★64 July 26, 2019 4:59 AM

Easy? Really?

9 Share Reply

dudelogin ★54 August 12, 2019 1:47 AM

Why is this an easy problem? It took me quite a while to wrap my head around this one.

4 Share Reply

aloginov ★153 December 26, 2018 8:49 AM

Almost every sentence of this solution explanation leaves some confusion.
For example how one should interpret this phrase: "This problem is about performing the steps directly as written."? (which steps? written where? no context given)
Or this: "There are some cases."? (what the author wants to say here? there are always *some* cases)
Or this: "In a more general setting, ..."? (setting? maybe case(s) instead of setting)

Read More

3 Share Reply

SHOW 1 REPLY

canadianczar ★126 February 4, 2020 2:52 AM

I feel that explaining this question on the whiteboard alone would take up the entire interview.

2 Share Reply

rahulthankachan ★19 April 14, 2019 6:24 AM

Cool Solution with the Greedy approach!!

0 Share Reply

zerustech ★200 February 19, 2019 10:22 AM

@awice Could you explain how to calculate the time complexity of this algorithm? Thanks.

0 Share Reply

ed2k ★1 September 25, 2018 12:13 AM

doesn't look correct, try this one,
"255.0.0.255"
256
seems n.bit_length() ==> n-#1<<(32-mask) is missing something.
for example, n=6, bit_length() 3, you just need mask=29 (3 bits to cover 6) however, your result is n-=4

Read More

0 Share Reply

sschangi ★183 April 22, 2018 1:13 AM

@awice What is a valid IPV4? because for e.g., the proposed alg, does not generate the correct answer for the test case: "0.0.0.0" and n = 4.

0 Share Reply

zhuangjianing ★5 March 26, 2020 8:03 AM

It's tough~

0 Share Reply