728 Self Dividing Numbers

2018年5月4日 12:52

A *self-dividing number* is a number that is divisible by every digit it contains.
For example, 128 is a self-dividing number because 128 % 1 == 0, 128 % 2 == 0, and 128 % 8 == 0.
Also, a self-dividing number is not allowed to contain the digit zero.
Given a lower and upper number bound, output a list of every possible self dividing number, including the bounds if possible.
Example 1:
Input:
left = 1, right = 22
Output: [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 22]
Note:

- The boundaries of each input argument are 1 <= left <= right <= 10000.

来自 <https://leetcode.com/problems/self-dividing-numbers/description/>
自除数 是指可以被它包含的每一位数除尽的数。
例如，128 是一个自除数，因为 128 % 1 == 0，128 % 2 == 0，128 % 8 == 0。
还有，自除数不允许包含 0 。
给定上边界和下边界数字，输出一个列表，列表的元素是边界（含边界）内所有的自除数。
示例 1：
输入：
上边界left = 1, 下边界right = 22
输出： [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 22]
注意：

- 每个输入参数的边界满足 1 <= left <= right <= 10000。

Solution for Python3:

```
class Solution1:
    def selfDividingNumbers(self, left, right):
        """
        :type left: int
        :type right: int
```

```python
        :rtype: List[int]
        """
        def isSelfDividingNumbers(num):
            n = num
            while n:
                t = n % 10
                if t == 0 or num % (n % 10):
                    return False
                n //= 10
            return True
        ans = []
        for i in range(left, right + 1):
            if isSelfDividingNumbers(i):
                ans.append(i)
        return ans

class Solution2:
    def selfDividingNumbers(self, left, right):
        """
        :type left: int
        :type right: int
        :rtype: List[int]
        """
        def self_dividing(n):
            for d in str(n):
                if d == '0' or n % int(d):
                    return False
            return True
        ans = []
        return list(filter(self_dividing, range(left, right + 1)))
        # for n in range(left, right + 1):
        #   if self_dividing(n):
        #       ans.append(n)
        # return ans
```

```cpp
class Solution {
public:
    vector<int> selfDividingNumbers(int left, int right) {
        vector<int> ans;
        for (int n = left; n <= right; n++) {
            if (self_dividing(n))
                ans.push_back(n);
        }
        return ans;
    }
    bool self_dividing(int n) {
        string s = to_string(n);
        for (int i = 0; i < s.length(); i++) {
            if (s[i] == '0' || (n % (s[i] - '0')))
                return false;
        }
        return true;
    }
};
```