2048. Next Greater Numerically Balanced Number

Medium | Topics | Companies | Hint

An integer `x` is **numerically balanced** if for every digit `d` in the number `x`, there are **exactly** `d` occurrences of that digit in `x`.

Given an integer `n`, return *the* ***smallest numerically balanced*** *number* ***strictly greater*** *than* `n`.

Example 1:

```
Input: n = 1
Output: 22
Explanation:
22 is numerically balanced since:
- The digit 2 occurs 2 times.
It is also the smallest numerically balanced number strictly greater
than 1.
```

Example 2:

```
Input: n = 1000
Output: 1333
Explanation:
1333 is numerically balanced since:
- The digit 1 occurs 1 time.
- The digit 3 occurs 3 times.
It is also the smallest numerically balanced number strictly greater
than 1000.
Note that 1022 cannot be the answer because 0 appeared more than 0
times.
```

Example 3:

```
Input: n = 3000
Output: 3133
Explanation:
3133 is numerically balanced since:
- The digit 1 occurs 1 time.
- The digit 3 occurs 3 times.
It is also the smallest numerically balanced number strictly greater
than 3000.
```

Constraints:

- $0 <= n <= 10^6$

Python:

```
class Solution:
    def generate(self, n: int, current: int, remaining: int, count: list[int]) -> int:
        if remaining == 0:
            if current > n and all(c == 0 or c == i for i, c in enumerate(count)):
                return current
            return 0

        result = 0
        for d in range(1, 10):
            if result == 0 and count[d] < d and d - count[d] <= remaining:
                count[d] += 1
                result = self.generate(n, current * 10 + d, remaining - 1, count)
                count[d] -= 1
        return result

    def nextBeautifulNumber(self, n: int) -> int:
        length = len(str(n))
        count = [0] * 10

        result = self.generate(n, 0, length, count)
        count = [0] * 10
        next_len_result = self.generate(0, 0, length + 1, count)
        if result == 0:
            result = next_len_result
        return result
```

JavaScript:

```
var nextBeautifulNumber = function(n) {
    function generate(n, current, remaining, count) {
        if (remaining === 0) {
            if (current > n) {
                for (let d = 1; d <= 9; d++) {
                    if (count[d] > 0 && count[d] !== d) return 0;
                }
                return current;
            }
            return 0;
        }
```

```
        let result = 0;
        for (let d = 1; d <= 9 && result === 0; d++) {
            if (count[d] < d && d - count[d] <= remaining) {
                count[d]++;
                result = generate(n, current * 10 + d, remaining - 1, count);
                count[d]--;
            }
        }
        return result;
    }

    const length = n.toString().length;
    let count = new Array(10).fill(0);

    let result = generate(n, 0, length, count);
    count.fill(0);
    const nextLenResult = generate(0, 0, length + 1, count);
    if (result === 0) result = nextLenResult;
    return result;
};
```

Java:

```
class Solution {
    long generate(long n, long current, long remaining, long[] count) {
        if (remaining == 0) {
            if (current > n) {
                for (int d = 1; d <= 9; d++) {
                    if (count[d] > 0 && count[d] != d) return 0;
                }
                return current;
            }
            return 0;
        }

        long result = 0;
        for (int d = 1; d <= 9 && result == 0; d++) {
            if (count[d] < d && d - count[d] <= remaining) {
                count[d]++;
                result = generate(n, current * 10 + d, remaining - 1, count);
                count[d]--;
            }
        }
        return result;
```

```
    }

    public int nextBeautifulNumber(int n) {
        String num = String.valueOf(n);
        long length = num.length();
        long[] count = new long[10];

        long result = generate(n, 0, length, count);
        java.util.Arrays.fill(count, 0);
        long nextLenResult = generate(0, 0, length + 1, count);
        if (result == 0) result = nextLenResult;
        return (int) result;
    }
}
```