3461. Check If Digits Are Equal in String After Operations I

Easy | Topics | Companies | Hint

You are given a string `s` consisting of digits. Perform the following operation repeatedly until the string has **exactly** two digits:

- For each pair of consecutive digits in `s`, starting from the first digit, calculate a new digit as the sum of the two digits **modulo** 10.
- Replace `s` with the sequence of newly calculated digits, *maintaining the order* in which they are computed.

Return `true` if the final two digits in `s` are the **same**; otherwise, return `false`.

Example 1:

> **Input:** `s = "3902"`
>
> **Output:** `true`
>
> **Explanation:**
>
> - Initially, `s = "3902"`
> - First operation:
>   - `(s[0] + s[1]) % 10 = (3 + 9) % 10 = 2`
>   - `(s[1] + s[2]) % 10 = (9 + 0) % 10 = 9`
>   - `(s[2] + s[3]) % 10 = (0 + 2) % 10 = 2`
>   - `s` becomes `"292"`
> - Second operation:
>   - `(s[0] + s[1]) % 10 = (2 + 9) % 10 = 1`
>   - `(s[1] + s[2]) % 10 = (9 + 2) % 10 = 1`
>   - `s` becomes `"11"`
> - Since the digits in `"11"` are the same, the output is `true`.

Example 2:

> **Input:** `s = "34789"`
>
> **Output:** `false`
>
> **Explanation:**
>
> - Initially, `s = "34789"`.
> - After the first operation, `s = "7157"`.
> - After the second operation, `s = "862"`.
> - After the third operation, `s = "48"`.
> - Since `'4' != '8'`, the output is `false`.

Constraints:

- `3 <= s.length <= 100`
- `s` consists of only digits.

Python:

```
class Solution(object):
    def hasSameDigits(self, s):
        i = 0
        res = ""
        while len(s) > 2 and i < len(s) - 1:
            res += str((int(s[i]) + int(s[i + 1])) % 10)
            i += 1
            if i == len(s) - 1:
                s = res
                i = 0
                res = ""
        return len(s) == 2 and s[0] == s[1]
```

JavaScript:

```
const getMod10 = (n, i) => {
    const fast5 = [
        [1,0,0,0,0],
        [1,1,0,0,0],
        [1,2,1,0,0],
        [1,3,3,1,0],
        [1,4,1,4,1]
    ];

    const xunzhi = [
        [0,6,2,8,4],
        [5,1,7,3,9]
    ];

    let mod2 = 1;
    let mod5 = 1;

    let a = n, b = i;
    while (a > 0 || b > 0) {
        const na = a & 1;
        const nb = b & 1;
        if (nb && !na) {
            mod2 = 0;
            break;
        }
        a >>= 1;
        b >>= 1;
    }

    a = n;
    b = i;
    while (a > 0 || b > 0) {
        const na = a % 5;
        const nb = b % 5;
        mod5 = (mod5 * fast5[na][nb]) % 5;
        a = (a / 5) | 0;
        b = (b / 5) | 0;
    }

    return xunzhi[mod2][mod5];
};
```

```
const hasSameDigits = s => {
    const n = s.length - 1;
    let left = 0;
    let right = 0;

    for (let i = 0; i <= n; i++) {
        const val = s.charCodeAt(i) - 48;
        if (i <= n - 1)
            left = (left + getMod10(n - 1, i) * val) % 10;
        if (i >= 1)
            right = (right + getMod10(n - 1, i - 1) * val) % 10;
    }

    return left === right;
};
```

Java:

```
class Solution {
    public boolean hasSameDigits(String s) {
        int n = s.length() - 1;
        int left = 0;
        int right = 0;

        for (int i = 0; i <= n; i++) {
            int val = s.charAt(i) - 48;
            if (i <= n - 1)
                left = (left + getMod10(n - 1, i) * val) % 10;
            if (i >= 1)
                right = (right + getMod10(n - 1, i - 1) * val) % 10;
        }

        return left == right;
    }

    private int getMod10(int n, int i) {
        int[][] fast5 = {
            {1,0,0,0,0},
            {1,1,0,0,0},
            {1,2,1,0,0},
            {1,3,3,1,0},
            {1,4,1,4,1}
        };
        int[][] xunzhi = {
```

```
            {0,6,2,8,4},
            {5,1,7,3,9}
        };

        int mod2 = 1;
        int mod5 = 1;

        int a = n, b = i;
        while (a > 0 || b > 0) {
            int na = a & 1;
            int nb = b & 1;
            if (nb == 1 && na == 0) {
                mod2 = 0;
                break;
            }
            a >>= 1;
            b >>= 1;
        }

        a = n;
        b = i;
        while (a > 0 || b > 0) {
            int na = a % 5;
            int nb = b % 5;
            mod5 = (mod5 * fast5[na][nb]) % 5;
            a /= 5;
            b /= 5;
        }

        return xunzhi[mod2][mod5];
    }
}
```