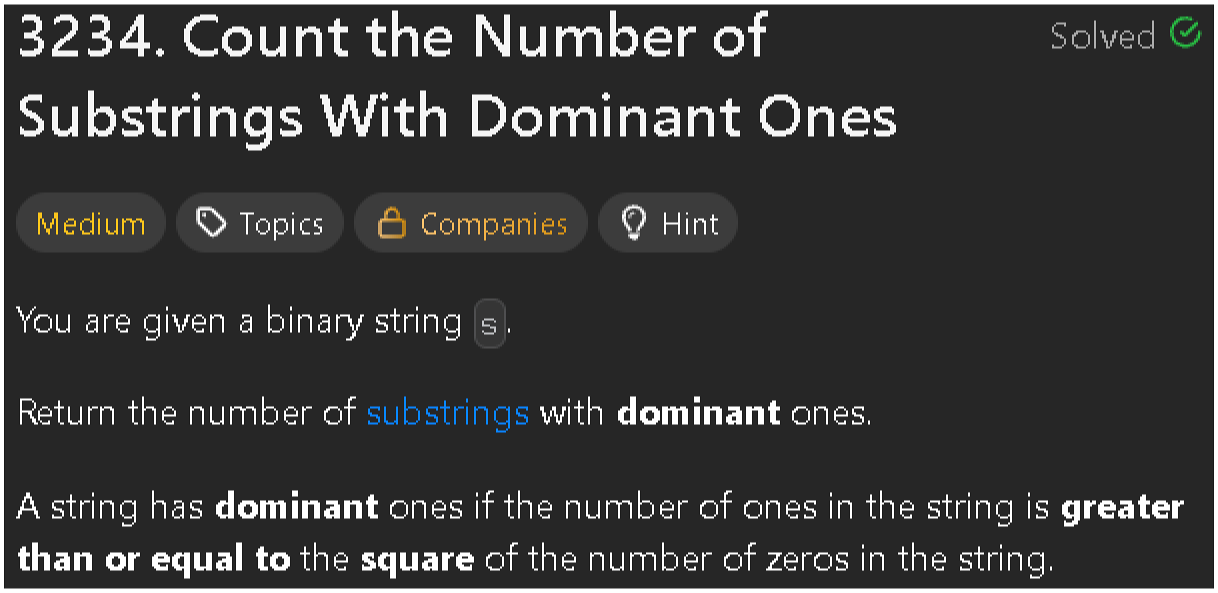


3234. Count the Number of Substrings With Dominant Ones

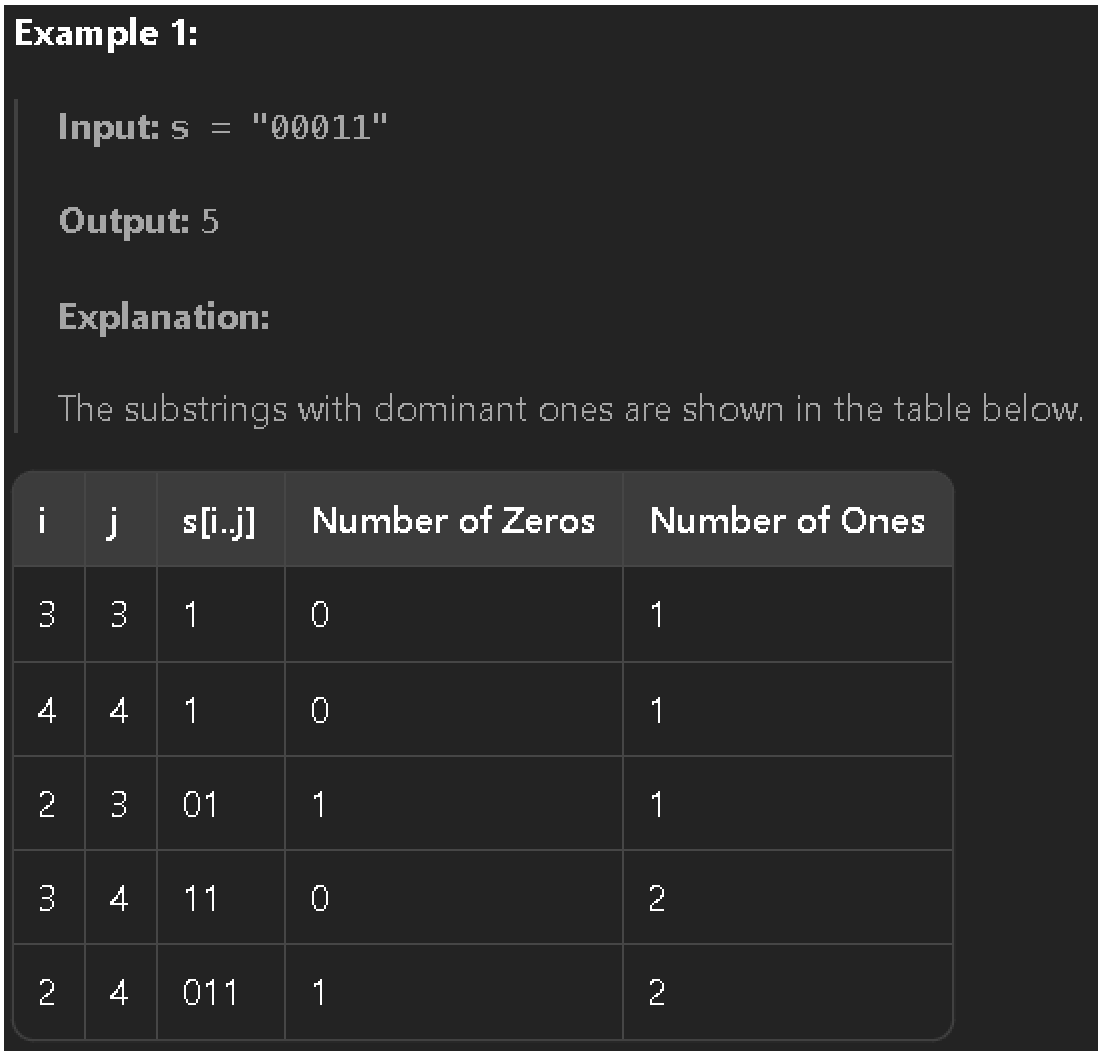
Solved ✓

Medium | Topics | Companies | Hint

You are given a binary string `s`.

Return the number of substrings with **dominant** ones.

A string has **dominant** ones if the number of ones in the string is **greater than or equal to** the **square** of the number of zeros in the string.



Example 1:

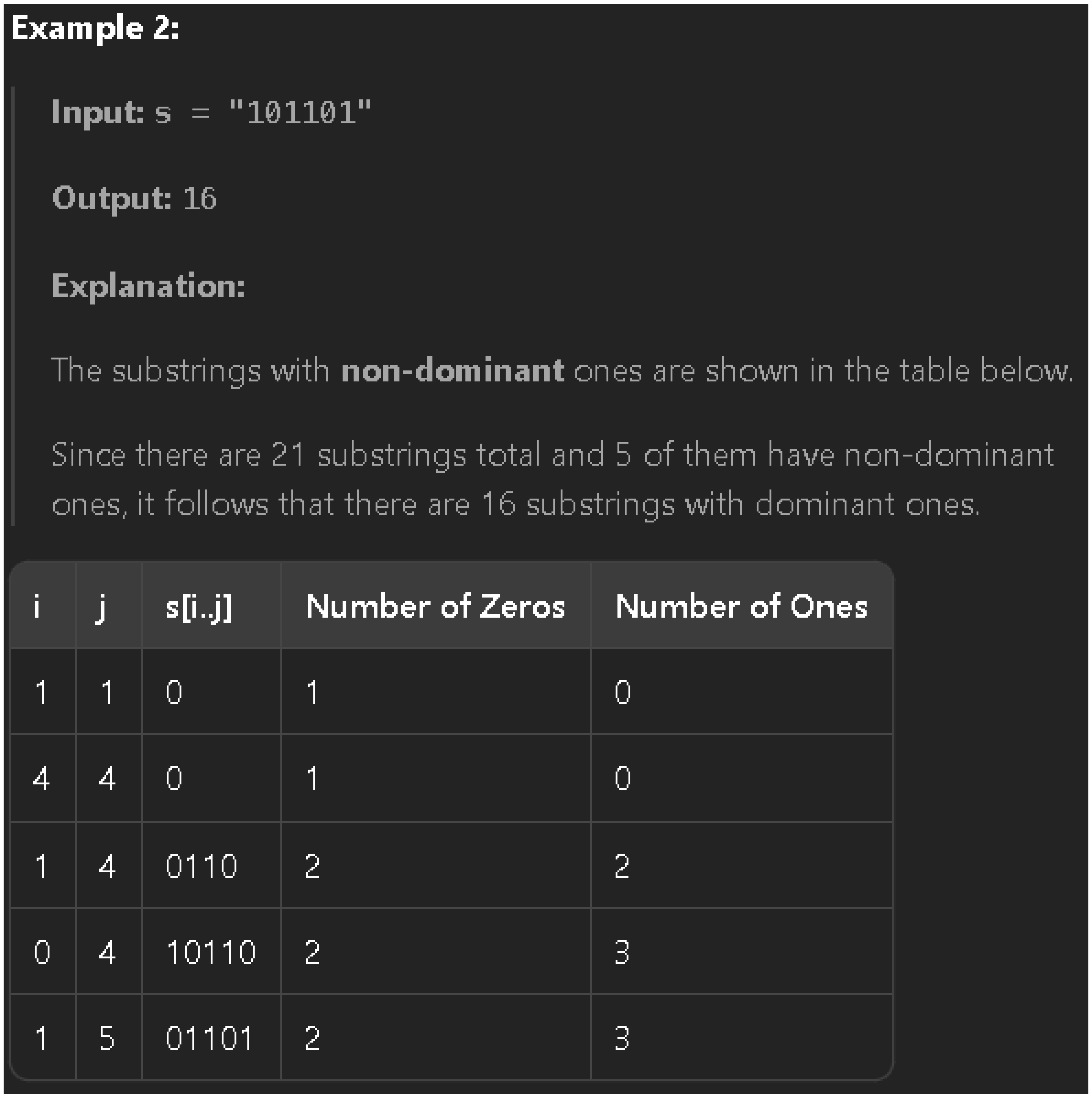
Input: `s = "00011"`

Output: 5

Explanation:

The substrings with dominant ones are shown in the table below.

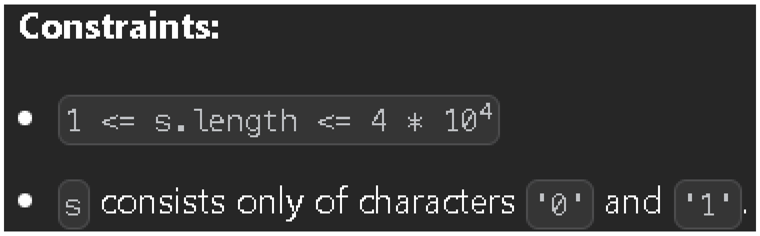
i	j	s[i..j]	Number of Zeros	Number of Ones
3	3	1	0	1
4	4	1	0	1
2	3	01	1	1
3	4	11	0	2
2	4	011	1	2



Example 2:

> **Input:** s = "101101"
>
> **Output:** 16
>
> **Explanation:**
>
> The substrings with **non-dominant** ones are shown in the table below.
>
> Since there are 21 substrings total and 5 of them have non-dominant ones, it follows that there are 16 substrings with dominant ones.

i	j	s[i..j]	Number of Zeros	Number of Ones
1	1	0	1	0
4	4	0	1	0
1	4	0110	2	2
0	4	10110	2	3
1	5	01101	2	3



Constraints:

- `1 <= s.length <= 4 * 10`^{4}
- `s` consists only of characters `'0'` and `'1'`.

Python:

```
class Solution:
    def numberOfSubstrings(self, s: str) -> int:
        n = len(s)
        pre = [-1] * (n + 1)
        for i in range(n):
            if i == 0 or s[i - 1] == "0":
                pre[i + 1] = i
            else:
                pre[i + 1] = pre[i]

        res = 0
        for i in range(1, n + 1):
            cnt0 = 1 if s[i - 1] == "0" else 0
            j = i
            while j > 0 and cnt0 * cnt0 <= n:
                cnt1 = (i - pre[j]) - cnt0
                if cnt0 * cnt0 <= cnt1:
                    res += min(j - pre[j], cnt1 - cnt0 * cnt0 + 1)
                j = pre[j]
                cnt0 += 1
        return res
```

JavaScript:

```
var numberOfSubstrings = function (s) {
    const n = s.length;
    const pre = new Array(n + 1);
    pre[0] = -1;
    for (let i = 0; i < n; i++) {
        if (i === 0 || (i > 0 && s[i - 1] === "0")) {
            pre[i + 1] = i;
        } else {
            pre[i + 1] = pre[i];
        }
    }
    let res = 0;
    for (let i = 1; i <= n; i++) {
        let cnt0 = s[i - 1] === "0" ? 1 : 0;
        let j = i;
        while (j > 0 && cnt0 * cnt0 <= n) {
            const cnt1 = i - pre[j] - cnt0;
            if (cnt0 * cnt0 <= cnt1) {
                res += Math.min(j - pre[j], cnt1 - cnt0 * cnt0 + 1);
            }
            j = pre[j];
            cnt0++;
        }
    }
    return res;
};
```

Java:

```
class Solution {

    public int numberOfSubstrings(String s) {
        int n = s.length();
        int[] pre = new int[n + 1];
        pre[0] = -1;
        for (int i = 0; i < n; i++) {
            if (i == 0 || (i > 0 && s.charAt(i - 1) == '0')) {
                pre[i + 1] = i;
            } else {
                pre[i + 1] = pre[i];
            }
        }
        int res = 0;
        for (int i = 1; i <= n; i++) {
            int cnt0 = s.charAt(i - 1) == '0' ? 1 : 0;
            int j = i;
            while (j > 0 && cnt0 * cnt0 <= n) {
                int cnt1 = (i - pre[j]) - cnt0;
                if (cnt0 * cnt0 <= cnt1) {
                    res += Math.min(j - pre[j], cnt1 - cnt0 * cnt0 + 1);
                }
                j = pre[j];
                cnt0++;
            }
        }
        return res;
    }
}
```