6023. Minimum White Tiles After Covering With Carpets

My Submissions (/contest/biweekly-contest-74/problems/minimum-white-tiles-after-covering-with-carpets/submissions/)

Back to Contest (/contest/biweekly-contest-74/)

You are given a **0-indexed binary** string `floor`, which represents the colors of tiles on a floor:

- `floor[i] = '0'` denotes that the i^{th} tile of the floor is colored **black**.
- On the other hand, `floor[i] = '1'` denotes that the i^{th} tile of the floor is colored **white**.

You are also given `numCarpets` and `carpetLen`. You have `numCarpets` **black** carpets, each of length `carpetLen` tiles. Cover the tiles with the given carpets such that the number of **white** tiles still visible is **minimum**. Carpets may overlap one another.

Return *the **minimum** number of white tiles still visible.*

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

Example 1:

```
Input: floor = "10110101", numCarpets = 2, carpetLen = 2
Output: 2
Explanation:
The figure above shows one way of covering the tiles with the carpets such that only 2 white tiles are visible.
No other way of covering the tiles with the carpets can leave less than 2 white tiles visible.
```

Example 2:

```
Input: floor = "11111", numCarpets = 2, carpetLen = 3
Output: 0
Explanation:
The figure above shows one way of covering the tiles with the carpets such that no white tiles are visible.
Note that the carpets are able to overlap one another.
```

Constraints:

- `1 <= carpetLen <= floor.length <= 1000`
- `floor[i]` is either `'0'` or `'1'`.
- `1 <= numCarpets <= 1000`

JavaScript

```
const MAX = Number.MAX_SAFE_INTEGER;
const initialize2DArray = (n, m) => { let d = []; for (let i = 0; i < n; i++) { let t = Array(m).fill(MAX);
d.push(t); } return d; };

const minimumWhiteTiles = (s, numCarpets, carpetLen) => {
    let n = s.length, dp = initialize2DArray(n + 1, numCarpets + 1);
    for (j = 0; j <= numCarpets; j++) dp[0][j] = 0;
    for (let i = 1; i <= n; i++) {
        for (let j = 0; j <= numCarpets; j++) {
            dp[i][j] = dp[i - 1][j] + (s[i - 1] - '0');
            if (j > 0) {
                dp[i][j] = Math.min(dp[i][j], dp[Math.max(0, i - carpetLen)][j - 1]);
            }
```

```
        }
    }
    return dp[n][numCarpets];
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/663155102/) | More Details > (/submissions/detail/663155102/)

Share your acceptance!