474. Ones and Zeroes

Solved

Medium | Topics | Companies

You are given an array of binary strings `strs` and two integers `m` and `n`.

Return *the size of the largest subset of* `strs` *such that there are* ***at most*** `m` `0`*'s and* `n` `1`*'s in the subset.*

A set `x` is a **subset** of a set `y` if all elements of `x` are also elements of `y`.

Example 1:

```
Input: strs = ["10","0001","111001","1","0"], m = 5, n = 3
Output: 4
Explanation: The largest subset with at most 5 0's and 3 1's
is {"10", "0001", "1", "0"}, so the answer is 4.
Other valid but smaller subsets include {"0001", "1"} and
{"10", "1", "0"}.
{"111001"} is an invalid subset because it contains 4 1's,
greater than the maximum of 3.
```

Example 2:

> **Input:** strs = ["10","0","1"], m = 1, n = 1
> **Output:** 2
> **Explanation:** The largest subset is {"0", "1"}, so the answer is 2.

Constraints:

- `1 <= strs.length <= 600`
- `1 <= strs[i].length <= 100`
- `strs[i]` consists only of digits `'0'` and `'1'`.
- `1 <= m, n <= 100`

Python:

```
class Solution:
    def findMaxForm(self, S: List[str], M: int, N: int) -> int:
        dp = [[0 for _ in range(N+1)] for _ in range(M+1)]
        for str in S:
            zeros = str.count("0")
            ones = len(str) - zeros
            for i in range(M, zeros - 1, -1):
                for j in range(N, ones - 1, -1):
                    dp[i][j] = max(dp[i][j], dp[i-zeros][j-ones] + 1)
        return dp[M][N]
```

JavaScript:

```
var findMaxForm = function(S, M, N) {
    let dp = Array.from({length:M+1},() => new Uint8Array(N+1))
    for (let i = 0; i < S.length; i++) {
        let str = S[i], zeros = 0, ones = 0
        for (let j = 0; j < str.length; j++)
            str.charAt(j) === "0" ? zeros++ : ones++
        for (let j = M; j >= zeros; j--)
            for (let k = N; k >= ones; k--)
                dp[j][k] = Math.max(dp[j][k], dp[j-zeros][k-ones] + 1)
    }
    return dp[M][N]
```

```
};
```

Java:

```
class Solution {
    public int findMaxForm(String[] S, int M, int N) {
        int[][] dp = new int[M+1][N+1];
        for (String str : S) {
            int zeros = 0, ones = 0;
            for (char c : str.toCharArray())
                if (c == '0') zeros++;
                else ones++;
            for (int i = M; i >= zeros; i--)
                for (int j = N; j >= ones; j--)
                    dp[i][j] = Math.max(dp[i][j], dp[i-zeros][j-ones] + 1);
        }
        return dp[M][N];
    }
}
```