Count Square Submatrices With All Ones

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	1277
tags	Dynamic Programming Prefix Sum
language	C++
solved on	@27/10/2024
link	https://leetcode.com/problems/count-square-submatrices-with-all-ones/description/
Completion	☑

Intuition

The task is to find the total number of square submatrices within a given binary matrix. A square submatrix can only exist where all elements in a contiguous square are 1s. We can leverage dynamic programming to keep track of the largest square submatrix ending at each cell, which allows us to count all possible squares efficiently.

Approach

1. Define a dynamic programming matrix `dp` where `dp[i][j]` represents the side length of the largest square submatrix that ends at cell `(i-1, j-1)` in the original matrix.
2. For each cell `(i, j)`, if `matrix[i-1][j-1] == 1`, calculate `dp[i][j]` as:

   $$\text{dp}[i][j] = 1 + \min(\text{dp}[i-1][j], \text{dp}[i][j-1], \text{dp}[i-1][j-1])$$

   This calculation considers the smallest square that can be formed by extending the square ending at the adjacent cells (top, left, and top-left) and adding the current cell to it.
3. Sum up the values in `dp` to get the total number of squares in the matrix since each `dp[i][j]` represents the count of squares with that cell as the bottom-right corner.
4. Return `totalSquares`, the sum of all `dp[i][j]` values.

Complexity

Time Complexity:

The time complexity is $O(m \times n)$ because we iterate through each cell of the matrix once.

Space Complexity:

The space complexity is $O(m \times n)$, as we use an additional `dp` matrix of the same size as the input matrix.

Code

```
int countSquares(vector<vector<int>>& matrix) {
    int m = matrix.size();
    int n = matrix[0].size();

    int totalSquares = 0;
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (matrix[i - 1][j - 1] == 1) {
                dp[i][j] = 1 + min({dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1]});
                totalSquares += dp[i][j];
            }
        }
    }

    return totalSquares;
}
```