1277. Count Square Submatrices with All Ones

Given a `m * n` matrix of ones and zeros, return how many **square** submatrices have all ones.

Example 1:

```
Input: matrix =
[
  [0,1,1,1],
  [1,1,1,1],
  [0,1,1,1]
]
Output: 15
Explanation:
There are 10 squares of side 1.
There are 4 squares of side 2.
There is  1 square of side 3.
Total number of squares = 10 + 4 + 1 = 15.
```

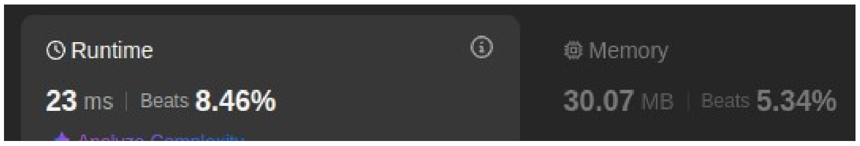
Example 2:

```
Input: matrix =
[
  [1,0,1],
  [1,1,0],
  [1,1,0]
]
Output: 7
Explanation:
There are 6 squares of side 1.
There is 1 square of side 2.
Total number of squares = 6 + 1 = 7.
```

Constraints:

- `1 <= arr.length <= 300`
- `1 <= arr[0].length <= 300`
- `0 <= arr[i][j] <= 1`

1277. Count Square Submatrices with All Ones


Runtime

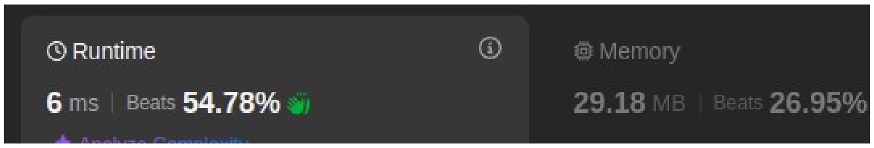
23 ms | Beats 8.46%

Memory

30.07 MB | Beats 5.34%

```
/*
   Recursion+Memoization
   Time compelxity: O(m.n)
   Space compelxity: O(m.n)
*/
class Solution {
   public:
      int countSquares(std::vector<std::vector<int>>& matrix) {
         int m=matrix.size();
         int n=matrix[0].size();

         std::vector<std::vector<int>> memo(m,std::vector<int>(n,-1));

         auto solve=[&](int row,int col,auto& self)->int{
            if(row==m || col==n || matrix[row][col]==0) return 0;
            if(memo[row][col]!=-1) return memo[row][col];
            return memo[row][col]=1+std::min
                                    ({
                                       self(row+1,col,self),
                                       self(row,col+1,self),
                                       self(row+1,col+1,self),
                                    });
         };

         int ans=0;
         for(int i=0;i<m;++i){
            for(int j=0;j<n;++j){
               ans+=solve(i,j,solve);
            }
         }
         return ans;
      }
};
```

1277. Count Square Submatrices with All Ones


Runtime

6 ms | Beats 54.78%

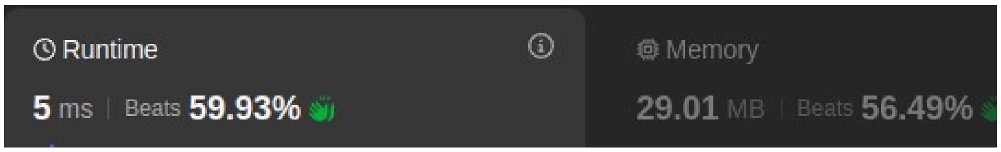
Memory

29.18 MB | Beats 26.95%

```
/*
   Bottom up with 2D array
   Time compelxity: O(m.n)
   Space compelxity: O(m.n)
*/
class Solution {
   public:
      int countSquares(std::vector<std::vector<int>>& matrix) {
         int m=matrix.size();
         int n=matrix[0].size();

         std::vector<std::vector<int>> dp(m+1,std::vector<int>(n+1,0));

         int ans=0;
         for(int row=1;row<=m;++row){
            for(int col=1;col<=n;++col){
               if(matrix[row-1][col-1]){
                  dp[row][col]=1+std::min
                                    ({
                                       dp[row-1][col],
                                       dp[row][col-1],
                                       dp[row-1][col-1]
                                    });
                  ans+=dp[row][col];
               }
            }
         }

         return ans;
      }
};
```

1277. Count Square Submatrices with All Ones

```
/*
   Bottom up with 2x1D array
   Time compelxity: O(m.n)
   Space compelxity: O(n)
*/
```



Runtime

5 ms | Beats 59.93%

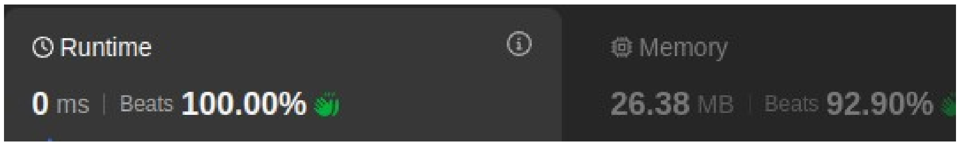
Memory

29.01 MB | Beats 56.49%

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

         std::vector<int> prev_row(n+1,0);

         int ans=0;
         for(int row=1;row<=m;++row){
            std::vector<int> cur_row(n+1,0);
            for(int col=1;col<=n;++col){
               if(matrix[row-1][col-1]==1){
                  cur_row[col]=1+std::min
                           ({
                              prev_row[col],
                              prev_row[col-1],
                              cur_row[col-1]
                           });
                  ans+=cur_row[col];
               }
            }
            prev_row=cur_row;
         }

         return ans;
      }
};
```

1277. Count Square Submatrices with All Ones

```
/*
   Bottom up without extra space
   Time compelxity: O(m.n)
   Space compelxity: O(1)
*/
```


Runtime

0 ms | Beats 100.00%

Memory

26.38 MB | Beats 92.90%

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

         int ans=std::count(matrix[0].begin(),matrix[0].end(),1);
         for(int row=1;row<m;++row) ans+=matrix[row][0];

         for(int row=1;row<m;++row){
            for(int col=1;col<n;++col){
               if(matrix[row][col]==1){
                  matrix[row][col]=1+std::min
                           ({
                              matrix[row-1][col],
                              matrix[row-1][col-1],
                              matrix[row][col-1]
                           });
                  ans+=matrix[row][col];
               }
            }
         }

         return ans;
      }
};
```