Gold Mine Problem

Given a gold mine called **M** of (**n x m)** dimensions. Each field in this mine contains a positive integer which is the amount of gold in tons. Initially the miner can start from any row in the first column. From a given cell, the miner can move

1. to the cell diagonally up towards the right
2. to the right
3. to the cell diagonally down towards the right

Find out maximum amount of gold which he can collect.

Example 1:

```
Input: n = 3, m = 3
M = {{1, 3, 3},
     {2, 1, 4},
     {0, 6, 4}};
Output: 12
Explaination:
The path is {(1,0) -> (2,1) -> (2,2)}.
```

Example 2:

```
Input: n = 4, m = 4
M = {{1, 3, 1, 5},
     {2, 2, 4, 1},
     {5, 0, 2, 3},
     {0, 6, 1, 2}};
Output: 16
Explaination:
The path is {(2,0) -> (3,1) -> (2,2)
-> (2,3)} or {(2,0) -> (1,1) -> (1,2)
-> (0,3)}.
```

Your Task:
You do not need to read input or print anything. Your task is to complete the function **maxGold()** which takes the values n, m and the mine M as input parameters and returns the maximum amount of gold that can be collected.

Expected Time Complexity: O(nm)

Expected Auxiliary Space: O(nm)

```
def maxGold(self, n, m, M):
        # code here
        dp = [[0]*m for _ in range(n)]
        for i in range(n):
            dp[i][0] = M[i][0]
        for j in range(1,m):
            for i in range(n):
                left_down = dp[i-1][j-1] if i-1>=0 else 0
                left_forward = dp[i][j-1]
                left_upward = dp[i+1][j-1] if i+1<n else 0
                dp[i][j] = M[i][j]+max(left_down,left_forward,left_upward)
        ans = 0
        for i in range(n):
            ans = max(ans,dp[i][m-1])
        return ans
```