1. Range Sum Query 2D - Immutable

Given a 2D matrix *matrix*, find the sum of the elements inside the rectangle defined by its upper left corner (*row*1, *col*1) and lower right corner (*row*2, *col*2).

Range Sum Query 2D The above rectangle (with the red border) is defined by (row1, col1) = **(2, 1)** and (row2, col2) = **(4, 3)**, which contains sum = **8**.

**Example:**

Given matrix = [  
 [3, 0, 1, 4, 2],  
 [5, 6, 3, 2, 1],  
 [1, 2, 0, 1, 5],  
 [4, 1, 0, 1, 7],  
 [1, 0, 3, 0, 5]  
]  
  
sumRegion(2, 1, 4, 3) -> 8  
sumRegion(1, 1, 2, 2) -> 11  
sumRegion(1, 2, 2, 4) -> 12

**Note:**

1. You may assume that the matrix does not change.
2. There are many calls to *sumRegion* function.
3. You may assume that *row*1 ≤ *row*2 and *col*1 ≤ *col*2.

**解** 同一维数组一样，先预处理

定义数组表示前 i-1 行前 j-1 列交叉区域的和

* 预处理阶段：
* 查询阶段：

class NumMatrix {  
public:  
 vector<vector<int>>S;  
 NumMatrix(vector<vector<int>>& matrix) {  
 int m = matrix.size();  
 if(m > 0){  
 int n = matrix[0].size();  
 S.resize(m+1, vector<int>(n+1, 0));  
 for(int i = 0; i < m; ++i){  
 for(int j = 0; j < n; ++j){  
 S[i+1][j+1] = matrix[i][j] + S[i][j+1]+S[i+1][j] - S[i][j];  
 }  
 }  
 }  
 }  
   
 int sumRegion(int row1, int col1, int row2, int col2) {  
 if(S.size() == 0)return 0;  
 return S[row2+1][col2+1] - S[row1][col2+1] - S[row2+1][col1] + S[row1][col1];  
 }  
};