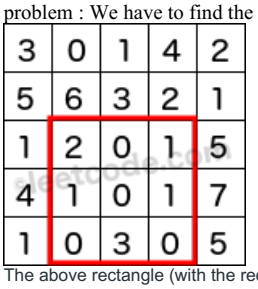
Range Sum Query 2D

---

problem : We have to find the sum of all elements in a rectangular protion of a 2d matrix



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

Naive Approach:

loop through (r1,c1) to (r2,c2) and get sum

problem with this aproach:
This approcah is good for one time calling but if we require Range sum multiple times then it is
better to store the results for future Uses.

Dynamic Programming Approach:

1. Store matrix sum from (0,0) to (r,c) at each index dp[r][c]
2. To Find dp[i][j] :

   dp[0][0] = mat[0][0]
   dp[i][0] = dp[i-1][0] + mat[i][0]
   dp[0][i] = dp[0][i-1] + mat[0][i]
   dp[i][j] = mat[i][j] + dp[i-1][j] + dp[i][j-1] - dp[i-1][j-1]
3. Now to Find required sum :
   sum += dp[r2][c2] - dp[r1-1][c2] - dp[r2][c1-1] + dp[r1-1][c1-1]

4. return sum

CODE :

```
class NumMatrix {
    vector<vector<int>> mat;
    int dp[999][999];
    int m,n;
public:
    NumMatrix(vector<vector<int>>& mat) {
        if(!mat.size() || !mat[0].size())
            return;
        m = mat.size();
        n = mat[0].size();
    dp[0][0] = mat[0][0];
        for(int i=1;i<m;i++)
            dp[i][0] = mat[i][0] + dp[i-1][0];
        for(int j=1;j<n;j++)
            dp[0][j] = mat[0][j] + dp[0][j-1];
        for(int i=1;i<m;i++)
            for(int j=1;j<n;j++)
                dp[i][j] = mat[i][j] + dp[i-1][j] + dp[i][j-1] - dp[i-1][j-1];
    }
    int sumRegion(int r1, int c1, int r2, int c2) {

        int res = dp[r2][c2];
        if(c1>0)
            res = res - dp[r2][c1-1];
        if(r1>0)
            res =  res - dp[r1-1][c2];
        if(r1>0  && c1>0)
            res = res + dp[r1-1][c1-1];
        return res;
    }
};
```