Submatrix Sum Queries

Given a matrix of size M x N, there are large number of queries to find submatrix sums. Inputs to queries are left top and right bottom indexes of submatrix whose sum is to find out.

How to preprocess the matrix so that submatrix sum queries can be performed in O(1) time.

Example:

```
tli :  Row number of top left of query submatrix
tlj :  Column number of top left of query submatrix
rbi :  Row number of bottom right of query submatrix
rbj :  Column number of bottom right of query submatrix

Input: mat[M][N] = {{1, 2, 3, 4, 6},
                    {5, 3, 8, 1, 2},
                    {4, 6, 7, 5, 5},
                    {2, 4, 8, 9, 4} };
Query1: tli = 0, tlj = 0, rbi = 1, rbj = 1
Query2: tli = 2, tlj = 2, rbi = 3, rbj = 4
Query3: tli = 1, tlj = 2, rbi = 3, rbj = 3;

Output:
Query1: 11  // Sum between (0, 0) and (1, 1)
Query2: 38  // Sum between (2, 2) and (3, 4)
Query3: 38  // Sum between (1, 2) and (3, 3)
```

We strongly recommend you to minimize your browser and try this yourself first.

The idea is to first create an **auxiliary matrix aux[M][N]** such that aux[i][j] stores sum of elements in submatrix from (0,0) to (i,j). Once aux[][] is constructed, we can compute sum of submatrix between (tli, tlj) and (rbi, rbj) in O(1) time. We need to consider aux[rbi][rbj] and subtract all unncessary elements. Below is complete expression to compute submatrix sum in O(1) time.

```
Sum between (tli, tlj) and (rbi, rbj) is,
   aux[rbi][rbj] - aux[tli-1][rbj] -
   aux[rbi][tlj-1] + aux[tli-1][tlj-1]

The submatrix aux[tli-1][tlj-1] is added because
elements of it are subtracted twice.
```

Illustration:

```
mat[M][N] = {{1, 2, 3, 4, 6},
             {5, 3, 8, 1, 2},
             {4, 6, 7, 5, 5},
             {2, 4, 8, 9, 4} };

We first preprocess the matrix and build
following aux[M][N]
aux[M][N] = {1,  3,   6,  10, 16}
            {6,  11,  22, 27,  35},
            {10, 21,  39, 49,  62},
            {12, 27,  53, 72,  89} }

Query : tli = 2, tlj = 2, rbi = 3, rbj = 4

Sum between (2, 2) and (3, 4) = 89 - 35 - 27 + 11
                              = 38
```

How to build aux[M][N]?

1. Copy first row of mat[][] to aux[][]
2. Do column wise sum of the matrix and store it.

3. Do the row wise sum of updated matrix aux[][] in step 2.

Below is C++ program based on above idea.

```
// C++ program to compute submatrix query sum in O(1)
// time
#include<iostream>
using namespace std;
#define M 4
#define N 5

// Function to preprcess input mat[M][N].  This function
// mainly fills aux[M][N] such that aux[i][j] stores sum
// of elements from (0,0) to (i,j)
int preProcess(int mat[M][N], int aux[M][N])
{
   // Copy first row of mat[][] to aux[][]
   for (int i=0; i<N; i++)
      aux[0][i] = mat[0][i];

   // Do column wise sum
   for (int i=1; i<M; i++)
      for (int j=0; j<N; j++)
         aux[i][j] = mat[i][j] + aux[i-1][j];

   // Do row wise sum
   for (int i=0; i<M; i++)
      for (int j=1; j<N; j++)
         aux[i][j] += aux[i][j-1];
}

// A O(1) time function to compute sum of submatrix
// between (tli, tlj) and (rbi, rbj) using aux[][]
// which is built by the preprocess function
int sumQuery(int aux[M][N], int tli, int tlj, int rbi,
                                              int rbj)
{
    // result is now sum of elements between (0, 0) and
    // (rbi, rbj)
    int res = aux[rbi][rbj];

    // Remove elements between (0, 0) and (tli-1, rbj)
    if (tli > 0)
       res = res - aux[tli-1][rbj];

    // Remove elements between (0, 0) and (rbi, tlj-1)
    if (tlj > 0)
       res = res - aux[rbi][tlj-1];

    // Add aux[tli-1][tlj-1] as elements between (0, 0)
    // and (tli-1, tlj-1) are subtracted twice
    if (tli > 0 && tlj > 0)
       res = res + aux[tli-1][tlj-1];

    return res;
}

// Driver program
int main()
{
   int mat[M][N] = {{1, 2, 3, 4, 6},
                    {5, 3, 8, 1, 2},
                    {4, 6, 7, 5, 5},
                    {2, 4, 8, 9, 4} };
   int aux[M][N];

   preProcess(mat, aux);

   int tli = 2, tlj = 2, rbi = 3, rbj = 4;
   cout << "\nQuery1: " << sumQuery(aux, tli, tlj, rbi, rbj);

   tli = 0, tlj = 0, rbi = 1, rbj = 1;
   cout << "\nQuery2: " << sumQuery(aux, tli, tlj, rbi, rbj);
```

```
    cout << "\nQuery2: " << sumQuery(aux, tli, tlj, rbi, rbj);

    tli = 1, tlj = 2, rbi = 3, rbj = 3;
    cout << "\nQuery3: " << sumQuery(aux, tli, tlj, rbi, rbj);

    return 0;
}
```

Output:

```
Query1: 38
Query2: 11
Query3: 38
```

Source: http://www.geeksforgeeks.org/amazon-interview-experience-set-241-1-5-years-experience/