Find a specific pair in Matrix

Given an n x n matrix mat[n][n] of integers, find the maximum value of mat(c, d) – mat(a, b) over all choices of indexes such that both c > a and d > b.

Example:

```
Input:
mat[N][N] = {{ 1, 2, -1, -4, -20 },
             { -8, -3, 4, 2, 1 },
             { 3, 8, 6, 1, 3 },
             { -4, -1, 1, 7, -6 },
             { 0, -4, 10, -5, 1 }};
Output: 18
The maximum value is 18 as mat[4][2]
- mat[1][0] = 18 has maximum difference.
```

The program should do only ONE traversal of the matrix. i.e. expected time complexity is $O(n^2)$

A **simple solution** would be to apply Brute-Force. For all values mat(a, b) in the matrix, we find mat(c, d) that has maximum value such that c > a and d > b and keeps on updating maximum value found so far. We finally return the maximum value.

Below is its implementation.

```
// A Naive method to find maximum value of mat1[d]
// - ma[a][b] such that c > a and d > b
#include <bits/stdc++.h>
using namespace std;
#define N 5

// The function returns maximum value A(c,d) - A(a,b)
// over all choices of indexes such that both c > a
// and d > b.
int findMaxValue(int mat[][N])
{
    // stores maximum value
    int maxValue = INT_MIN;

    // Consider all possible pairs mat[a][b] and
    // mat1[d]
    for (int a = 0; a < N - 1; a++)
      for (int b = 0; b < N - 1; b++)
         for (int c = a + 1; c < N; c++)
           for (int d = b + 1; d < N; d++)
              if (maxValue < (mat1[d] - mat[a][b]))
                  maxValue = mat1[d] - mat[a][b];

    return maxValue;
}

// Driver program to test above function
int main()
{
   int mat[N][N] = {
                   { 1, 2, -1, -4, -20 },
                   { -8, -3, 4, 2, 1 },
                   { 3, 8, 6, 1, 3 },
                   { -4, -1, 1, 7, -6 },
                   { 0, -4, 10, -5, 1 }
                 };
    cout << "Maximum Value is "
         << findMaxValue(mat);

    return 0;
}
```

Output:

```
Maximum Value is 18
```

The above program runs in O(n^4) time which is nowhere close to expected time complexity of O(n^2)

An **efficient solution** uses extra space. We pre-process the matrix such that index(i, j) stores max of elements in matrix from (i, j) to (N-1, N-1) and in the process keeps on updating maximum value found so far. We finally return the maximum value.

```
// An efficient method to find maximum value of mat1[d]
// - ma[a][b] such that c > a and d > b
#include <bits/stdc++.h>
using namespace std;
#define N 5

// The function returns maximum value A(c,d) - A(a,b)
// over all choices of indexes such that both c > a
// and d > b.
int findMaxValue(int mat[][N])
{
    //stores maximum value
    int maxValue = INT_MIN;

    // maxArr[i][j] stores max of elements in matrix
    // from (i, j) to (N-1, N-1)
    int maxArr[N][N];

    // last element of maxArr will be same's as of
    // the input matrix
```

```
    maxArr[N-1][N-1] = mat[N-1][N-1];

    // preprocess last row
    int maxv = mat[N-1][N-1];  // Initialize max
    for (int j = N - 2; j >= 0; j--)
    {
        if (mat[N-1][j] > maxv)
            maxv = mat[N - 1][j];
        maxArr[N-1][j] = maxv;
    }

    // preprocess last column
    maxv = mat[N - 1][N - 1];  // Initialize max
    for (int i = N - 2; i >= 0; i--)
    {
        if (mat[i][N - 1] > maxv)
            maxv = mat[i][N - 1];
        maxArr[i][N - 1] = maxv;
    }

    // preprocess rest of the matrix from bottom
    for (int i = N-2; i >= 0; i--)
    {
        for (int j = N-2; j >= 0; j--)
        {
            // Update maxValue
            if (maxArr[i+1][j+1] - mat[i][j] >
                                            maxValue)
                maxValue = maxArr[i + 1][j + 1] - mat[i][j];

            // set maxArr (i, j)
            maxArr[i][j] = max(mat[i][j],
                               max(maxArr[i][j + 1],
                                   maxArr[i + 1][j]) );
        }
    }

    return maxValue;
}

// Driver program to test above function
int main()
{
    int mat[N][N] = {
                       { 1, 2, -1, -4, -20 },
                       { -8, -3, 4, 2, 1 },
                       { 3, 8, 6, 1, 3 },
                       { -4, -1, 1, 7, -6 },
                       { 0, -4, 10, -5, 1 }
                     };
    cout << "Maximum Value is "
         << findMaxValue(mat);

    return 0;
}
```

Output:

```
Maximum Value is 18
```

If we are allowed to modify of the matrix, we can avoid using extra space and use input matrix instead.

Exercise: Print index (a, b) and (c, d) as well.