二维度区间查询（矩阵）

```
#include<bits/stdc++.h>
using namespace std;
/*
 *  二维RMQ,预处理复杂度 n*m*log*(n)*log(m)
 *  数组下标从1开始
 */
int val[310][310];
int dp[310][310][9][9]; //  最大值
int mm[310];            //  二进制位数减一,使用前初始化

void initRMQ(int n, int m)
{
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
        {
            dp[i][j][0][0] = val[i][j];
        }
    }
    for (int ii = 0; ii <= mm[n]; ii++)
    {
        for (int jj = 0; jj <= mm[m]; jj++)
        {
            if (ii + jj)
            {
                for (int i = 1; i + (1 << ii) - 1 <= n; i++)
                {
                    for(int j = 1; j + (1 << jj) - 1 <= m; j++)
                    {
                        if (ii)
                        {
                            dp[i][j][ii][jj] = max(dp[i][j][ii - 1][jj], dp[i +
(1 << (ii - 1))][j][ii - 1][jj]);
                        }
                        else
                        {
                            dp[i][j][ii][jj] = max(dp[i][j][ii][jj - 1], dp[i][j
+ (1 << (jj - 1))][ii][jj - 1]);
                        }
                    }
                }
            }
        }
    }
}

//  查询矩形内的最大值(x1<=x2,y1<=y2)
int rmq(int x1, int y1, int x2, int y2)
{
    int k1 = mm[x2 - x1 + 1];
    int k2 = mm[y2 - y1 + 1];
    x2 = x2 - (1 << k1) + 1;
    y2 = y2 - (1 << k2) + 1;
    return max(max(dp[x1][y1][k1][k2], dp[x1][y2][k1][k2]), max(dp[x2][y1][k1]
[k2], dp[x2][y2][k1][k2]));
}

int main()
{
    //  在外面对mm数组进行初始化
    mm[0] = -1;
    for (int i = 1; i <= 305; i++)
    {
        mm[i] = ((i & (i - 1)) == 0) ? mm[i - 1] + 1 : mm[i - 1];
    }
    int n, m;
    int Q;
    int r1, c1, r2, c2;
    while (scanf("%d%d", &n, &m) == 2)
    {
        for (int i = 1; i <= n; i++)
        {
            for (int j = 1; j <= m; j++)
            {
                scanf("%d", &val[i][j]);
            }
        }
        initRMQ(n, m);
        scanf("%d", &Q);
        while(Q--)
        {
            scanf("%d%d%d%d", &r1, &c1, &r2, &c2);
            if (r1 > r2)
            {
                swap(r1, r2);
            }
            if (c1 > c2)
            {
                swap(c1, c2);
            }
            int tmp = rmq(r1, c1, r2, c2);
            printf("%d ", tmp);
            if (tmp == val[r1][c1] || tmp == val[r1][c2] || tmp == val[r2][c1]
|| tmp == val[r2][c2])
            {
                printf("yes\n");
            }
            else
            {
                printf("no\n");
            }
        }
    }
    return 0;
}
```