子矩阵修改问题

问题描述

给定一个 $n \times m$ 大小的矩阵 A。

给定 q 组操作，每次操作为给定 5 个正整数 x_1, y_1, x_2, y_2, d，A_{x_1,y_1} 是子矩阵左上角端点，A_{x_2,y_2} 是子矩阵右下角端点，你需要给其中每个元素都增加 d。

输出操作结束后的矩阵 A。

输入格式

第一行输入 3 个正整数 n, m, q。 $(1 \le n, m \le 10^3, 1 \le q \le 10^5)$

接下来 n 行每行输入 m 个整数，表示 $A_{i,j}$。$(-10^3 \le A_{i,j} \le 10^3, 1 \le i \le n, 1 \le j \le m)$

接下来 q 行，每行输入 5 个正整数 x_1, y_1, x_2, y_2, d。
$(1 \le x_1 \le x_2 \le n, 1 \le y_1 \le y_2 \le m, -10^3 \le d \le 10^3)$

输出格式

输出 n 行 m 个整数，表示操作结束后的矩阵 A。

暴力

暴力实现该问题的时间复杂度为 $O(q \times n \times m)$。

二维差分

明确一点:

前缀和数组求差分后是原数组，原数组求差分后是差分数组。

差分数组求前缀和是原数组，原数组求前缀和后是前缀和数组。

- C++

```cpp
#include <iostream>
using namespace std;

const int N = 1005;
int a[N][N], b[N][N];

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);

    int n, m, q;
    cin >> n >> m >> q;

    // 读取初始矩阵
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            cin >> a[i][j];
        }
    }

    // 计算初始差分数组
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            b[i][j] = a[i][j] - a[i - 1][j] - a[i][j - 1] + a[i - 1][j - 1];
        }
    }

    // 处理 q 次增量修改
    while (q--) {
        int x1, y1, x2, y2, c;
        cin >> x1 >> y1 >> x2 >> y2 >> c;
        b[x1][y1] += c;
        b[x2 + 1][y1] -= c;
        b[x1][y2 + 1] -= c;
        b[x2 + 1][y2 + 1] += c;
    }

    // 还原前缀和得到最终矩阵
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            a[i][j] = b[i][j] + a[i - 1][j] + a[i][j - 1] - a[i - 1][j - 1];
        }
    }

    // 输出最终矩阵
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            cout << a[i][j] << " ";
        }
        cout << "\n";
    }

    return 0;
}

```

- Java

```java
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        PrintWriter out = new PrintWriter(System.out);

        // 读取矩阵大小 n, m 和查询次数 q
        int n = in.nextInt(), m = in.nextInt(), q = in.nextInt();

        // 定义矩阵 a (存储初始值) 和 差分数组 b
        int[][] a = new int[n + 1][m + 1];
        int[][] b = new int[n + 1][m + 1];

        // 读取初始矩阵值
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                a[i][j] = in.nextInt();
            }
        }

        // 计算初始差分数组
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                b[i][j] = a[i][j] - a[i - 1][j] - a[i][j - 1] + a[i - 1][j - 1];
            }
        }

        // 处理 q 次增量修改
        while (q-- > 0) {
            int x1 = in.nextInt(), y1 = in.nextInt(), x2 = in.nextInt(), y2 = in.nextInt();
            int c = in.nextInt();
            b[x1][y1] += c;
            b[x2 + 1][y1] -= c;
            b[x1][y2 + 1] -= c;
            b[x2 + 1][y2 + 1] += c;
        }

        // 还原前缀和得到最终矩阵
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                a[i][j] = b[i][j] + a[i - 1][j] + a[i][j - 1] - a[i - 1][j - 1];
            }
        }

        // 输出最终矩阵
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                out.print(a[i][j] + " ");
            }
            out.println();
        }

        out.flush();
        in.close();
    }
}

```

- Python

```
import sys

def main():
    # 读取输入
    n, m, q = map(int, sys.stdin.readline().split())

    # 读取初始矩阵
    a = [[0] * (m + 2) for _ in range(n + 2)]
```

```python
    b = [[0] * (m + 2) for _ in range(n + 2)]

    for i in range(1, n + 1):
        row = list(map(int, sys.stdin.readline().split()))
        for j in range(1, m + 1):
            a[i][j] = row[j - 1]

    # 计算初始差分数组
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            b[i][j] = a[i][j] - a[i - 1][j] - a[i][j - 1] + a[i - 1][j - 1]

    # 处理 q 次修改
    for _ in range(q):
        x1, y1, x2, y2, c = map(int, sys.stdin.readline().split())
        b[x1][y1] += c
        b[x2 + 1][y1] -= c
        b[x1][y2 + 1] -= c
        b[x2 + 1][y2 + 1] += c

    # 还原前缀和得到最终矩阵
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            a[i][j] = b[i][j] + a[i - 1][j] + a[i][j - 1] - a[i - 1][j - 1]

    # 输出最终矩阵
    for i in range(1, n + 1):
        print(" ".join(map(str, a[i][1:m+1])))

if __name__ == "__main__":
    main()

```

棋盘

- C++

```cpp
#include <iostream>
using namespace std;

const int N = 2005;
int a[N][N], b[N][N];

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);

    int n, m;
    cin >> n >> m ;

    // 处理 q 次增量修改
    while (m--) {
        int x1, y1, x2, y2, c;
        cin >> x1 >> y1 >> x2 >> y2;
        b[x1][y1] += 1;
        b[x2 + 1][y1] -= 1;
        b[x1][y2 + 1] -= 1;
        b[x2 + 1][y2 + 1] += 1;
    }

    // 还原前缀和得到最终矩阵
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            a[i][j] = b[i][j] + a[i - 1][j] + a[i][j - 1] - a[i - 1][j - 1];
            if(a[i][j]%2==1){
              cout<<1;
            }else{
              cout<<0;
            }
        }
        cout<<"\n";
    }

    return 0;
}

```

- Java

```
import java.util.Scanner;

public class Main {
    static final int N = 2005;
    static int[][] a = new int[N][N], b = new int[N][N];

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int m = scanner.nextInt();

        while (m-- > 0) {
            int x1 = scanner.nextInt();
            int y1 = scanner.nextInt();
            int x2 = scanner.nextInt();
            int y2 = scanner.nextInt();
            b[x1][y1] += 1;
            b[x2 + 1][y1] -= 1;
            b[x1][y2 + 1] -= 1;
            b[x2 + 1][y2 + 1] += 1;
        }

        // 还原前缀和得到最终矩阵
        for (int i = 1; i <= n; ++i) {
            for (int j = 1; j <= n; ++j) {
                a[i][j] = b[i][j] + a[i - 1][j] + a[i][j - 1] - a[i - 1][j - 1];
                System.out.print(a[i][j] % 2 == 1 ? 1 : 0);
            }
            System.out.println();
        }

        scanner.close();
    }
}

```

- Python

```python
import sys

N = 2005
a = [[0] * N for _ in range(N)]
b = [[0] * N for _ in range(N)]

def main():
    input = sys.stdin.read
    data = input().split()
    index = 0

    n, m = int(data[index]), int(data[index + 1])
    index += 2

    for _ in range(m):
        x1, y1, x2, y2 = map(int, data[index:index + 4])
        index += 4
        b[x1][y1] += 1
        b[x2 + 1][y1] -= 1
        b[x1][y2 + 1] -= 1
        b[x2 + 1][y2 + 1] += 1

    # 还原前缀和得到最终矩阵
    output = []
    for i in range(1, n + 1):
        row = []
        for j in range(1, n + 1):
            a[i][j] = b[i][j] + a[i - 1][j] + a[i][j - 1] - a[i - 1][j - 1]
            row.append('1' if a[i][j] % 2 == 1 else '0')
        output.append("".join(row))

    print("\n".join(output))

if __name__ == "__main__":
    main()

```