【宫水三叶】下次如何在 30 秒内做出来？二维前缀和模板如何记忆

宫水三叶
发布于 2021-03-02 8.1k 数组 Java 前缀和

前缀和解法（二维）

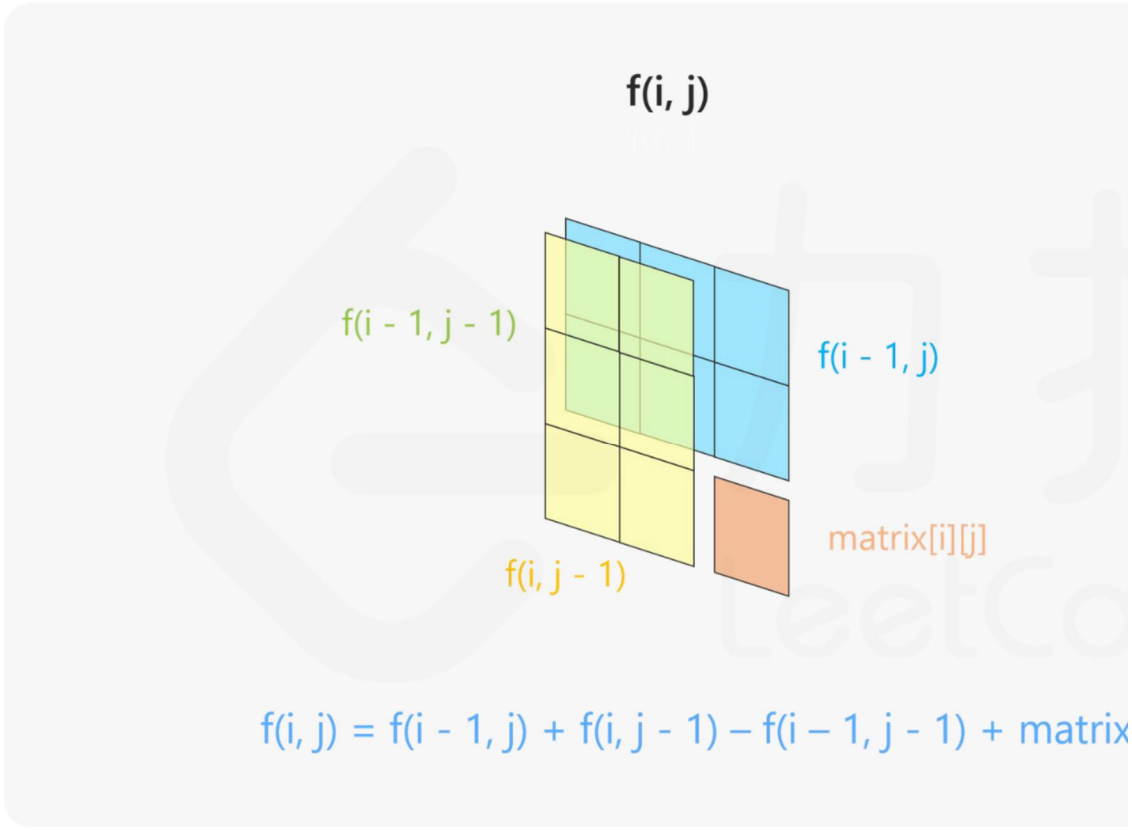
真就是今天的「每日一题」呗 ~

这是一道「二维前缀和」的裸题。

「二维前缀和」解决的是二维矩阵中的矩形区域求和问题。

二维前缀和数组中的每一个格子记录的是「以当前位置为区域的右下角（区域左上角恒定为原数组的左上角）的区域和」

贴一张官解示意图，我觉得很清晰：



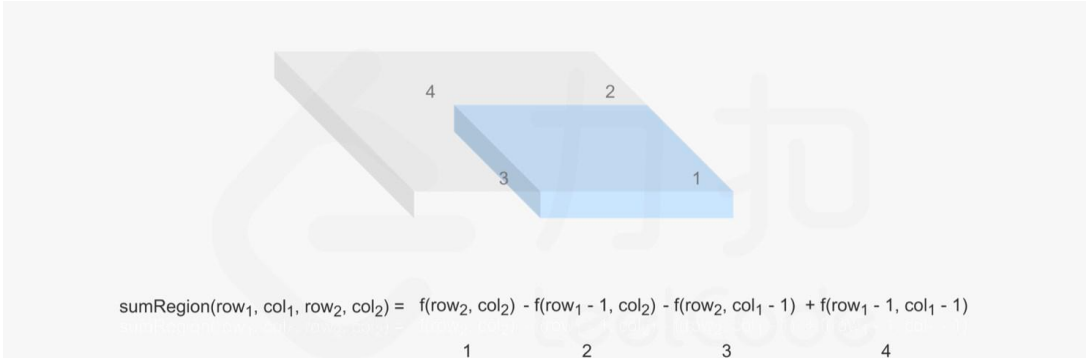
如果觉得不清晰，请将 f[i][j] 理解成是以 (i, j) 为右下角，(0, 0) 为左上角的区域和。

因此当我们要求 (x1, y1) 作为左上角，(x2, y2) 作为右下角 的区域和的时候，可以直接利用前缀和数组快速求解：

```
sum[x2][y2] - sum[x1 - 1][y2] - sum[x2][y1 - 1] + sum[x1 - 1][y1 - 1]
```

再贴官解示意图（画图困难户在学做图了，别骂😭）：

(x1, y1) == (row1, col1)，(x2, y2) == (row2, col2)



代码（感谢@Qian 同学提供的 C++ 代码）：

- Java
- C++

```
class NumMatrix {
    int[][] sum;
    public NumMatrix(int[][] matrix) {
        int n = matrix.length, m = n == 0 ? 0 : matrix[0].length;
        // 与「一维前缀和」一样，前缀和数组下标从 1 开始，因此设定矩阵形状为 [n + 1][m + 1]（模板部分）
        sum = new int[n + 1][m + 1];
        // 预处理除前缀和数组（模板部分）
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                sum[i][j] = sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j - 1] + matrix[i - 1][j - 1];
            }
        }
    }

    public int sumRegion(int x1, int y1, int x2, int y2) {
        // 求某一段区域和 [i, j] 的模板是 sum[x2][y2] - sum[x1 - 1][y2] - sum[x2][y1 - 1] + sum[x1 - 1][y1 - 1];（模板部分）
        // 但由于我们源数组下标从 0 开始，因此要在模板的基础上进行 + 1
        x1++; y1++; x2++; y2++;
        return sum[x2][y2] - sum[x1 - 1][y2] - sum[x2][y1 - 1] + sum[x1 - 1][y1 - 1];
    }
}
```

```
class NumMatrix {
public:
    vector<vector<int>> sum;
    NumMatrix(vector<vector<int>>& matrix) {
        int n = matrix.size(), m = n == 0 ? 0 : matrix[0].size();
        // 与「一维前缀和」一样，前缀和数组下标从 1 开始，因此设定矩阵形状为 [n + 1][m + 1]（模板部分）
        sum.resize(n+1, vector<int>(m+1,0));
        // 预处理除前缀和数组（模板部分）
        for(int i = 1; i <= n; i++) {
            for(int j = 1; j <= m; j++) {
                sum[i][j] = sum[i-1][j] + sum[i][j-1] - sum[i-1][j-1] + matrix[i-1][j-1];
            }
        }
    }

    int sumRegion(int x1, int y1, int x2, int y2) {
        // 求某一段区域和 [i, j] 的模板是 sum[x2][y2] - sum[x1 - 1][y2] - sum[x2][y1 - 1] + sum[x1 - 1][y1 - 1];（模板部分）
        // 但由于我们源数组下标从 0 开始，因此要在模板的基础上进行 + 1
        x1++; y1++; x2++; y2++;
        return sum[x2][y2] - sum[x1-1][y2] - sum[x2][y1-1] + sum[x1-1][y1-1];
    }
};
```

- 时间复杂度：预处理前缀和数组需要对原数组进行线性扫描，复杂度为 $O(n*m)$，计算结果复杂度为 $O(1)$。整体复杂度为 $O(n*m)$
- 空间复杂度：$O(n*m)$

二维前缀和模板【重点】

下面重点分享下前缀和模板该如何记忆，其实很多模板都可以按照类似方式进行记忆。

虽然「二维前缀和」的模板相比「一维前缀和」的模板要长，但是逻辑是清晰的，可以现场推导。

当然也可以**在理解的基础上**，使用以下方式进行记忆：

- Java
- C++

```
// 预处理前缀和数组
{
    sum = new int[n + 1][m + 1];
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            // 当前格子（和） = 上方的格子（和） + 左边的格子（和） - 左上角的格子（和） + 当前格子（值）【和是指对应的前缀和，值是指原数组中的值】
            sum[i][j] = sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j - 1] + matrix[i - 1][j - 1];
        }
    }
}

// 首先我们要令左上角为 (x1, y1) 右下角为 (x2, y2)
// 计算 (x1, y1, x2, y2) 的结果
{
    // 前缀和是从 1 开始，原数组是从 0 开始，上来先将原数组坐标全部 +1，转换为前缀和坐标
    x1++; y1++; x2++; y2++;
    // 记作 22 - 12 - 21 + 11，然后 不减，减第一位，减第二位，减两位
    // 也可以记作 22 - 12(x - 1) - 21(y - 1) + 11(x y 都 - 1)
    ans = sum[x2][y2] - sum[x1 - 1][y2] - sum[x2][y1 - 1] + sum[x1 - 1][y1 - 1];
}
```

```
// 预处理前缀和数组
{
    sum.resize(n+1, vector<int>(m+1,0));
    // 预处理除前缀和数组（模板部分）
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= m; j++) {
            // 当前格子（和） = 上方的格子（和） + 左边的格子（和） - 左上角的格子（和） + 当前格子（值）【和是指对应的前缀和，值是指原数组中的值】
            sum[i][j] = sum[i-1][j] + sum[i][j-1] - sum[i-1][j-1] + matrix[i-1][j-1];
        }
    }
}
```

```
// 首先我们要令左上角为 (x1, y1) 右下角为 (x2, y2)
// 计算 (x1, y1, x2, y2) 的结果
{
    // 前缀和是从 1 开始，原数组是从 0 开始，上来先将原数组坐标全部 +1，转换为前缀和坐标
    x1++; y1++; x2++; y2++;
    // 记作 22 - 12 - 21 + 11，然后 不减，减第一位，减第二位，减两位
    // 也可以记作 22 - 12(x - 1) - 21(y - 1) + 11(x y 都 - 1)
    ans = sum[x2][y2] - sum[x1-1][y2] - sum[x2][y1-1] + sum[x1-1][y1-1];

}
```

猜题小游戏（2021-03-02）

哈哈哈哈，昨天蒙中了。今天再蒙一道明天的「每日一题」~

与其说是猜题小游戏，不如说是拓展性的类似题目推荐：

307. 区域和检索 - 数组可修改

大家可以试试做 ~

老规矩，无论是不是明天的每日一题，我都会在明天发布该题相关题解 ~

（2021-03-03）开奖结果：未中奖，谢谢参与。以下是相关题解：

关于各类「区间和」问题如何选择解决方案（含模板）...

最后

如果有帮助到你，请给个点赞关注，让更多的人看到 ~ (ˉ▽ˉ)/

也欢迎你 关注我，和三叶一起刷穿 LeetCode ~

提供讲「证明」&「思路」的题解

下一篇：【带图示】Java 二维前缀和

52

条评论~

最热

谢先 登录 后发表评论

yancey

2021-04-22

没太想明白预处理部分最后不应该是matrix[i][j]么？为什么要减一了呢？

3 踩 查看 9 条回复 收起回复 回复 分享 举报

刷题の好孩子 L1

2021-04-22

谢谢三叶老师，学会啦~2333

2 踩 查看 1 条回复 收起回复 回复 分享 举报

ther

L3

2021-03-02

感觉这题不需要模板吧，草稿纸上画画图基本就出来了

3 踩 查看 7 条回复 收起回复 回复 分享 举报