

**《计算机算法课程实践》**

**实验报告**

**实验（ 四 ）**

**年 级： 2024**

**专 业： 计算机技术**

**姓 名： 刘兴宇**

**学 号： 2120240739**

# 1 实验内容

带颜色约束的数字菱形求解。

在严格对称的数字菱形中（总行数为奇数），每个节点的构成为（颜色，数值），其中颜色​为：红（R）、绿（G）、蓝（B），​数值​为整数（可正可负），从顶部出发，每一步可移动到下一行的左下方或右下方节点，且相邻节点颜色必须不同。求合法路径的最大数值和，并输出具体路径。

# 2 设计思想

本问题是典型的求解最优化问题的模型，具有明显的阶段性，符合动态规划的应用场景。

**2.1 算法选择**

选择动态规划是因为本问题具备以下两个核心特性：

1. **最优子结构**：一个节点的最优路径（最大和）必然建立在其某个父节点的最优路径之上。也就是说，到达节点 (i, j) 的最大路径和，等于其父节点 (i-1, k) 的最大路径和，加上节点 (i, j) 本身的数值。
2. **重叠子问题**：在计算过程中，同一个节点的父节点的最优解会被其多个子节点重复使用。动态规划通过填表的方式，将子问题的解存储起来，避免了重复计算。

**2.2 状态定义与转移**

* **状态定义**：我们定义一个二维数组 dp[i][j]，表示从菱形顶部出发，遵循移动规则到达第 i 行第 j 个节点时，能够获得的最大路径数值和。
* **状态转移方程**： 要计算 dp[i][j]，我们需要考察其所有合法的父节点。由于菱形的几何结构，父节点的位置在菱形的扩张部分和收缩部分有所不同。
  + **对于菱形上半部分（扩张阶段）**：节点 (i, j) 的父节点为 (i-1, j-1) 和 (i-1, j)。
  + **对于菱形下半部分（收缩阶段）**：节点 (i, j) 的父节点为 (i-1, j) 和 (i-1, j+1)。

状态转移方程为： dp[i][j] = diamond[i][j].value + max(dp\_from\_valid\_parents) 其中，dp\_from\_valid\_parents 是所有满足**颜色不同**约束条件的父节点的 dp 值中的最大者。如果不存在合法的父节点，则该路径中断。

**2.3 路径回溯**

为了输出具体路径，仅有 dp 表是不够的。我们引入一个辅助的二维数组 path[i][j]，在计算 dp[i][j] 时，将贡献了最大和的那个父节点的**列索引**存入 path[i][j]。

计算完成后，整个问题的最优解即为最后一个节点 dp[n-1][0] 的值。从终点 (n-1, 0) 开始，利用 path 表反向查找，逐层向上回溯，即可重建整个最优路径。

3 程序效果

程序的运行效果，输入及输出的相关要求和具体执行结果如下所示：

## 3.1 输入

Total number of rows in the diamond (n): 7

Enter the diamond nodes (color and value) for each row, separated by spaces:

R10

G-5 B8

R2 G20 B1

G3 B-10 R15 G6

G0 B5 R8

B4 R-2

G9

## 3.2 输出

Max path sum: 65

Path colors sequence: R -> B -> G -> R -> B -> R -> G

Path values sequence: 10 -> 8 -> 20 -> 15 -> 5 -> -2 -> 9

# 4 算法分析

 **时间复杂度：O(n²)** 程序的核心计算是填充 dp 表和 path 表。我们需要遍历菱形中的每一个节点一次。对于一个 n 行的菱形，总节点数 N 的数量级为 O(n²)。对每个节点，我们执行常数次（最多两次）的父节点比较操作。因此，填充表格的时间复杂度为 O(n²)。路径回溯的过程需要从最后一行走到第一行，步数为 n，其时间复杂度为 O(n)。综上，算法的总体时间复杂度为 O(n²)。

 **空间复杂度：O(n²)** 算法需要额外的空间来存储菱形的节点数据、dp 表以及 path 表。这三个数据结构的大小都与菱形中的总节点数成正比，即 O(n²)。因此，算法的空间复杂度为 O(n²)。

# 5 总结

本次实验通过动态规划方法，成功解决了数字菱形中的最大路径和问题。通过设计 dp 状态和状态转移方程，有效地处理了路径选择中的最优决策，并利用 path 表实现了具体路径的回溯。