# 运筹学实验报告-2

PB21000231 刘舟洋

## 一、问题背景

本项目研究了最短路径问题的求解方法，比较了专用的 **Dijkstra 算法** 和通用的 **线性规划方法（LP）** 在随机生成的连通图上的性能。实验测试了不同规模的图，发现 Dijkstra 算法在运行时间上显著优于 LP 方法，尤其在大规模图中表现突出，而 LP 方法更适合解决带有复杂约束的优化问题。两种方法在结果上完全一致，为选择最优求解策略提供了数据支持。

## 二、程序介绍

1. Dijkstra 算法

**算法目标：**找到从源点 s 到目标点 t 的最短路径。

**基本思想：**

- Dijkstra 算法是一种贪心算法，专为解决非负权重图的单源最短路径问题。
- 通过逐步扩展从源点到其他节点的最短路径，确保每次选取的是距离源点最近的节点进行更新。

**实现过程：**

1. 使用一个优先队列（在代码中使用 heapq 模块实现）来存储当前节点的最短路径。
2. 初始化源点的距离为 0，其他所有节点的距离为无穷大。
3. 每次从优先队列中弹出距离最小的节点，更新其相邻节点的距离。
4. 重复以上步骤，直到所有节点的最短路径都被确定，或者到达目标点 t。
5. 返回源点到目标点的最短路径距离。

时间复杂度：O((V + E) \log V)，其中 V 是节点数，E 是边数。

**2. 线性规划（LP）方法**

**算法目标：**通过线性规划模型求解最短路径问题。

**基本思想：**

- 将最短路径问题转化为一个流量优化问题，通过线性规划求解。
- 设计目标函数和约束条件，使得优化结果对应最短路径的总权重。

**模型构建：**

1. **变量**：为每条边定义流量变量 x(i, j)，表示是否选择这条边。
2. **目标函数**：最小化路径总权重（所有边权重与对应流量的乘积之和）。
3. **约束条件**：

- 流量平衡约束：确保源点的净流出为 1，目标点的净流入为 1，其余节点流入等于流出。
- 非负约束：每条边的流量非负。

**实现过程：**

1. 使用 pulp 库构建线性规划模型。
2. 添加目标函数和约束条件。
3. 调用 CBC 求解器进行求解。
4. 提取并返回最优解的目标值，即最短路径的总权重。

**时间复杂度**：取决于求解器的实现，一般为多项式时间，与问题规模（变量数和约束数）相关。

# 三、实验结果

| 节点数 | 边数 | Dijkstra结果 | Dijkstra时间 | LP结果 | LP时间 | 一致性 |
|---|---|---|---|---|---|---|
| 1000 | 3745 | 49 | 0.0022s | 49.0 | 0.1486 | 是 |
| 5000 | 23328 | 50 | 0.0151s | 50.0 | 0.9019 | 是 |
| 10000 | 50528 | 37 | 0.0142s | 37.0 | 2.1489 | 是 |

### 1. 正确性

实验表明，Dijkstra 算法和 LP 方法在求解最短路径问题时均能获得一致的结果。这表明两种方法在处理非负权图时均具有很高的正确性和可靠性，能够准确地找到源点到目标点的最短路径。

### 2. 性能对比

Dijkstra 算法在所有测试中表现出显著的性能优势。即使在大规模图（如 10000 个节点和 50000 条边）上，运行时间仍然保持在毫秒级别，效率极高。而 LP 方法的运行时间随着图规模的增加迅速增长，从 1000 个节点的 0.1486 秒增加到 10000 个节点的 2.1489 秒，性能远不如 Dijkstra 算法。这表明 Dijkstra 算法更加适合解决大规模图的最短路径问题。

### 3. 时间复杂度趋势

Dijkstra 算法在节点数和边数增加时，运行时间增长较慢甚至趋于平稳，这得益于其专为最短路径设计的高效性。而 LP 方法的运行时间对图的节点数和边数更为敏感，随着图的规模迅速增加，模型的变量和约束数量也成倍增长，导致求解时间显著上升。

**4. 适用场景**

对于标准的最短路径问题（无负权图），Dijkstra 算法是最优选择，特别是在大规模稀疏图上表现尤为突出。而 LP 方法虽然效率不如 Dijkstra，但它具有通用性，适合处理带有复杂约束（如路径容量限制、节点访问顺序等）的最短路径问题。

# 四、总结

本实验通过随机生成不同规模的连通图，比较了 Dijkstra 算法 和 线性规划方法 (LP) 在求解最短路径问题上的性能。实验结果表明，Dijkstra 算法在运行效率上显著优于 LP 方法，特别是在大规模图中，其时间始终保持在毫秒级别。而 LP 方法尽管效率较低，但通用性更强，适用于带有复杂约束的优化问题。两种方法的求解结果在所有测试中完全一致，为不同需求场景下选择合适的算法提供了明确参考。