**摘要**

本报告介绍了一个基于 Python 和 PyQt 开发的路由算法图形化模拟器。该模拟器允许用户创建网络拓扑，选择并运行不同的路由算法，并可视化地展示路由选择过程。本报告详细说明了所选路由算法的基本原理，对性能进行了初步分析，并探讨了设计和实现的思路。

**1. 引言**

在网络技术中，路由算法是确定数据包从源到目的地最有效路径的关键组成部分。有效的路由算法可以提高网络的效率和可靠性。本项目旨在开发一个模拟器，以教育和研究的目的，对比和分析不同的路由算法。

**2. 路由算法的基本原理**

**2.1 Dijkstra 算法**

Dijkstra 算法是一种用于在加权图中找到最短路径的算法。它逐步扩展已知的最短路径，直到达到目的地。该算法适用于没有负权重边的图。

* **原理**：从起始节点开始，逐渐“放松”（更新）到达每个节点的最短路径估计，直到找到目的地的最短路径。
* **性能**：时间复杂度为 O(V^2)，其中 V 是节点的数量。使用优先队列可以降低到 O(V + E log V)。

**2.2 Bellman-Ford 算法**

Bellman-Ford 算法也用于找到加权图中的最短路径，但与 Dijkstra 算法不同，它可以处理具有负权重边的图。

* **原理**：通过重复放松图中的所有边，更新到达每个节点的最短路径估计。它还能检测负权重循环。
* **性能**：时间复杂度为 O(VE)，其中 V 是节点的数量，E 是边的数量。

**3. 性能指标分析**

性能分析主要集中在算法的时间复杂度和空间复杂度上。Dijkstra 算法在使用优先队列的情况下提供了更好的性能，特别是在稀疏图中。相比之下，Bellman-Ford 算法的性能较低，但它能处理更复杂的图结构，如包含负权重边的图。

**4. 设计思路与实现**

**4.1 设计思路**

本项目的设计理念是提供一个直观、用户友好的界面，让用户能够轻松地创建网络拓扑、选择和运行路由算法，并可视化结果。

**4.2 实现**

* **技术栈**：使用 Python 语言和 PyQt 框架进行开发，利用 NetworkX 库处理图和网络算法，Matplotlib 库进行图形的绘制。
* **用户界面**：提供交互式界面，允许用户输入节点和边，选择算法，并显示图形化的网络拓扑和路径。
* **算法实现**：Dijkstra 和 Bellman-Ford 算法被手动实现并集成到模拟器中。

**4.3 分析与计算**

* **算法选择**：根据网络的特性（如是否有负权重边）和性能需求来选择合适的算法。
* **路径可视化**：通过突出显示选定路径上的节点和边，直观地展示了算法的结果。

**5. 结论**

本模拟器成功实现了两种主要的路由算法，并提供了一种直观的方式来教学和研究网络路由。尽管存在性能限制，但该工具在教育和研究领域具有潜在的应用价值。