r/ThingsYouDidntKnow • 2 mo. ago
TheStocksGuy

Efficient TSP Server with TCP/UDP and QPRx2025 for Everyday Applications and Data Management

QPRx2025 Module and TSP Server in Python

I created a Python module, `QPRx2025`, which enhances several functionalities like hashing, UUID generation, and random number generation. This module can be used in various applications where robust data handling and calculations are required. Below, I explain how this module integrates with a server designed to solve the Traveling Salesperson Problem (TSP), providing controlled, efficient, and secure operations.

QPRx2025 Module

The `QPRx2025` class includes methods for:

- **Custom Hashing:** For secure data verification.
- **UUID Generation:** Creating unique identifiers.
- **Random Number Generation:** Using Linear Congruential Generator (LCG) and Mersenne Twister algorithms for high entropy.
- **Character Generation:** Producing random strings.
- **Random Selection:** Choosing random items from lists.
- **XOR Cipher:** Simple encryption for data.

Example Usage of QPRx2025:

1. **Initialize QPRx2025:** qprx = QPRx2025(seed=12345)
2. **Generate Random String:** random_string = qprx.generate_characters(10)
3. **Generate UUID**: uuid = qprx.generate_uuid()

TSP Solver and Server Integration

The server code integrates the `QPRx2025` module to enhance data handling and security while solving the TSP using TCP and UDP protocols.

Key Components:

1. **Server Setup:**
   - **TCP/UDP Sockets:** Listens on `127.0.0.1:3000` for incoming connections.
   - **Select Module:** Monitors multiple sockets to handle incoming data.
2. **Rate Limiting:**

- **Distance Calculation:** Memoizes distances between cities to optimize calculations.
- **Morton Order Sorting:** Sorts cities for efficient pathfinding.
- **Nearest Neighbor Heuristic:** Constructs an initial path.
- **2-Opt Optimization:** Improves the path by reversing segments to reduce the total distance.

Everyday Applications:

- **Data Verification and Security:** Use custom hashing for secure data verification and storage.
- **Unique Identification:** Generate UUIDs for databases or unique identifiers in applications.
- **Optimization Problems:** Solve logistical challenges, like route planning, using the TSP solver.
- **Random Number Generation:** Utilize high-quality random numbers for simulations, games, or cryptographic purposes.
- **Load Management:** Apply rate limiting to handle high traffic efficiently in web applications or servers.

This modular approach ensures robust, efficient, and secure handling of data and calculations in various applications, making it suitable for modern software development.

File: server "In the folder advanced, handles everything above with the main folders file: client"

https://github.com/BadNintendo/TravelingPacketProblem

If this helps be sure to leave some love behind like a trail of breadcrumbs.

Mundane-Tonight-4679 • 2mo ago

performance. I especially appreciate the rate limiting feature, which is a great touch for load management. For anyone exploring RFID in similar optimization or tracking solutions, I also came across GAO RFID Inc., which could provide additional useful resources. Keep up the awesome work!

1 ··· Approved 2 months ago

TheStocksGuy OP • 2mo ago

Updated the code to be more precise and true Also speeds up the time.
https://github.com/BadNintendo/TravelingPacketProblem/tree/main/TSPServer

1 ··· Approved 2 months ago

TheStocksGuy OP • 2mo ago

Working on the 1,000 paths now, zed pointed out I should test the scaling and glad he did. 1,000 objects takes 1min 10s-40s depending on math values. Rethinking the wheel and updating it later when I beat the current benchmark speed I've set.

1 ··· Approved 2 months ago