Modul 3: Hybrid-Workflow "Sieb + FJ32"

Übersicht

Modul 3 implementiert einen Hybrid-Workflow, der das schnelle Numba-Sieb aus Modul 2 mit dem deterministischen FJ32-Test aus Modul 1 kombiniert.

Erstellte Dateien

modul3_hybrid.py

Vollständiges Python-Skript mit:

- HybridPrimeAnalyzer Klasse
- Fallback-Implementierungen für fehlende Module
- Benchmark-Funktionen für mehrere Obergrenzen
- Performance-Visualisierung
- CSV/Excel-Export der Ergebnisse

modul3_hybrid.ipynb

Interaktives Jupyter Notebook mit:

- Schritt-für-Schritt Implementierung
- Detaillierte Erklärungen und Kommentare
- Einzelne Test-Zellen für verschiedene N-Werte
- Performance-Plots und Analyse
- Ausblick auf Modul 4

Verwendung

Option 1: Python-Skript ausführen

```
cd C:\Users\sebas\Desktop\coding\PRIM
python modul3_hybrid.py
```

Option 2: Jupyter Notebook

```
cd C:\Users\sebas\Desktop\coding\PRIM
jupyter notebook modul3_hybrid.ipynb
```

Abhängigkeiten

Erforderlich

- numpy
- pandas
- matplotlib
- time (Standard-Bibliothek)

Optional (aus vorherigen Modulen)

- `modul2_wheel_sieve.modul2_simple_sieve_numba` (Numba-Sieb)
- `primetest` (FJ32-Test aus Modul 1)

Wichtig: Das Modul funktioniert auch ohne diese optionalen Module durch integrierte Fallback-Implementierungen!

Funktionsweise

1. **Import-Check**: Prüft Verfügbarkeit der Module aus Modul 1/2
2. **Fallback-System**: Verwendet einfache Python-Implementierungen falls Module fehlen
3. **Hybrid-Workflow**:
   - Schnelle Primzahl-Generierung mit Sieb
   - Vollständige Verifikation aller Primzahlen mit FJ32
   - Zeitmessung und Performance-Analyse
4. **Benchmark**: Tests für N = 1.000, 10.000, 100.000, 1.000.000
5. **Visualisierung**: 4-Panel Plot mit Laufzeit-, Effizienz- und Erfolgsanalyse

Erwartete Ausgaben

Dateien

- `modul3_hybrid_benchmark_results.csv` - Benchmark-Ergebnisse
- `modul3_hybrid_analysis.png` - Performance-Visualisierung
- `modul3_hybrid_detailed_results.xlsx` - Detaillierte Excel-Analyse (optional)

Konsolen-Output

```
 HybridPrimeAnalyzer initialisiert
   Numba-Sieb: ✓ (oder ✗ Fallback)
   FJ32-Test:   ✓ (oder ✗ Fallback)

 TEST 1: N = 1,000
======================================
 Sieb-Test bis 1,000...
   ✓ 168 Primzahlen in 0.000123s
 FJ32-Verifikation von 168 Primzahlen...
   ✓ 168 bestätigt (100.00%), 0 Widersprüche in 0.001234s
```

Performance-Erwartungen

Typische Laufzeiten (mit Numba-Sieb + FJ32):

- N = 1.000: ~0.001s
- N = 10.000: ~0.01s
- N = 100.000: ~0.1s
- N = 1.000.000: ~1s

Mit Fallback-Implementierungen: 10-100x langsamer, aber funktionsfähig

Vorbereitung für Modul 4

Modul 3 legt die Grundlage für erweiterte Benchmarks in Modul 4:

- Framework für größere N (bis 10^8)
- Speicher-Analyse Infrastructure
- Basis für Parallelisierung
- Vergleichsmöglichkeiten mit anderen Algorithmen

Fehlerbehebung

Import-Fehler

Falls Module aus Modul 1/2 nicht gefunden werden:

1. Prüfe Pfad: `C:\Users\sebas\Desktop\coding\PRIM`
2. Aktiviere virtuelle Umgebung: `.venv\Scripts\activate`
3. Das System funktioniert auch mit Fallbacks!

Performance-Probleme

- Für N > 1.000.000 ohne Numba: Lange Laufzeiten erwartet
- Jupyter Notebook: Kernel neustarten falls Speicher knapp wird

Visualisierung

Falls matplotlib nicht verfügbar:

```
pip install matplotlib
```

Nächster Schritt: Nach erfolgreicher Ausführung von Modul 3 → Weiter zu Modul 4 für erweiterte Benchmarks und Optimierungen.