Analyse der Matrixmultiplikation und Matrixaddition

1.) Die Laufzeit:

Die Laufzeit $T(n)$ hängt von der Anzahl der Operationen ab, die für die Additionen und Multiplikationen der Matrizen erforderlich sind.
Die innere Schleife führt n Multiplikationen/Additionen aus, daher $O(n^3)$.

2.) Algorithmus Analyse:

```
for (int i = 0; i < n; i++)  {
    final int row = i;
    threads[i] = new Thread(() -> {
            for (int j = 0; j < n; j++) {
                int sum = 0;
                for (int k = 0; k < n; k++) {
                    sum += A[row][k] + B[k][j];
                }
                result[row][j] = sum;
            }
    });
     threads[i].start();
}
```

Hauptschleife des Algorithmus
Die äußere Schleife läuft n Mal
Die mittlere Schleife läuft n Mal
Die innere Schleife läuft n Mal

3.) Abschätzung in O-Notation:

Die Anzahl der Opertionen ist demnach $T(n) = O(n^3)$.

4.) Komplexitätsklasse:

Der Algorithmus liegt in der Komplexitätsklasse $O(n^3)$.

6.) Bestimmung der Konstanten:

Gemessene Laufzeiten:
$n = 1250: T(500) = 8467ms$
$n = 2500: T(2500) = 147191ms$

Bestimmung der Konstanten:

$$T(n) = O(n^3)$$

Das es Konstanten c_1 und c_2 gibt, sodass $c_2 * n^3 \leq T(n) \leq c_1 * n^3$ für $n \geq n_0$

Da in jeder der n^3 Iterationen eine konstante Anzahl an Operationen erfolgt:

$$T(n) \in \Theta(n^3)$$

Für $n = 1250$:
$T(1250) = 1250$
$1250^3 = 1953125000$
$c_1 * n^3 \leq T(n) \leq c_2 * n^3 \approx 1250$
$c_1 * 2000000000 \geq 1250^3 \rightarrow c_1 \leq \frac{1250}{1953125000} \approx 1.953125 * 10^9$
$c_2 * 1900000000 \geq 1250^3 \rightarrow c_2 \geq \frac{1250}{1953125000} \approx 1.953125 * 10^9$

Analyse der Strassen-Matrixmultiplikation

Die Strassen-Matrixmultiplikation ist ein divide-and-conquer Algorithmus, der die Anzahl der Multiplikationen im Vergleich zur klassischen Matrixmultiplikation reduziert. Er ist insbesondere für mittelgroße bis große quadratische Matrizen effizient.

Algorithmusbeschreibung

Gegeben zwei Matrizen A und B der Größe n×n, wobei n eine Zweierpotenz ist, wird jede Matrix in vier Untermatrizen der Größe n/2×n/2 geteilt. Es werden sieben Produkte rekursiv berechnet, die anschließend kombiniert werden, um das Endergebnis zu erhalten. Die Anzahl der Multiplikationen reduziert sich auf 7 (im Gegensatz zu 8 bei klassischer Methode).

Komplexitätsanalyse

Die rekursive Struktur der Strassen-Multiplikation folgt dem Rekursionsschema:

T(n) = 7T(n/2) + O(n^2)

Laut dem Master-Theorem ergibt sich daraus eine Zeitkomplexität von:

$$T(n) \in O(n^{\log_2(7)}) \approx O(n^{2.81})$$

Vergleich mit klassischer Multiplikation

Klassische Matrixmultiplikation hat die Komplexität O(n^3). Durch Strassen reduziert sich diese auf O(n^2.81), was bei großen Matrizen signifikante Leistungsverbesserungen bringt. Bei kleinen Matrizen kann der Overhead jedoch zu einer Verlangsamung führen.

Parallelisierung

Die rekursiven Aufrufe von Strassen lassen sich parallel ausführen. Durch den Einsatz von Threads bzw. ExecutorService können die sieben Teilprodukte gleichzeitig berechnet werden.

Fazit

Die Strassen-Multiplikation bietet durch ihre reduzierte Anzahl an Multiplikationen eine bessere theoretische Laufzeit als die klassische Methode. In der Praxis ist sie bei größeren Matrizen durch Parallelisierung effizient nutzbar.