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