Aufgabe 1:

Betrachten Sie die folgenden Codeausschnitte und bestimmen Sie die Zeitkomplexität in O(f(n))-Notation:

1.1

```
void codeA(int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i * i; j++) {
            tuwas();
        }
    }
}
```

1.2

```
void codeB(int n) {
    for (int i = 1; i <= n; i = i * 2) {
        for (int j = 1; j <= n; j++) {
            tuwas();
        }
    }
}
```

1.3

```
void codeC(int n) {
    int i = n;
    while (i > 0) {
        tuwas();
        i = i / 3;
    }
}
```

1.4

```
void codeD(int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j *= 2) {
            tuwas();
        }
    }
}
```

1.5

```
void codeE(int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = n; j > 0; j /= 2) {
            tuwas();
        }
    }
}
```

1.6

```
void codeF(int n) {
    for (int i = 1; i <= n; i *= 3) {
        tuwas();
    }
}
```

Aufgabe 2:

2.1 Gegeben sind die folgenden Funktionen:

1. **f(n) = n^2 + 4n + 7**
2. **g(n) = 5n^2 + 2n**
3. **h(n) = 3n^3 + 20n**

Bestimmen Sie die asymptotische Komplexität für jede Funktion in O(f(n))-Notation.

2.2 Betrachten Sie die folgenden Funktionen:

1. **f(n) = 2n^3 + 5n^2 + 10n + 15**
2. **g(n) = 7n^3 + 3n^2**
3. **h(n) = 100n^2 + 30n + 1000**

Bestimmen Sie die asymptotische Komplexität für jede Funktion in O(f(n))-Notation.

Aufgabe 3:

3.1 Betrachten Sie die folgenden Algorithmen und ihre Zeitkomplexität:

1. **A1: f1(n) = 3n + 5**
2. **A2: f2(n) = n^2 + 2n**
3. **A3: f3(n) = 2^n**

Bestimmen Sie, welcher Algorithmus am schnellsten ist, abhängig von der Größe von n. Vergleichen Sie die Algorithmen paarweise und lösen Sie die entsprechenden Ungleichungen, um die Schnittpunkte zu finden.

3.2 Gegeben sind die folgenden Algorithmen und ihre Zeitkomplexität:

1. **A1: f1(n) = n^2 + 5n + 10**
2. **A2: f2(n) = n log n + 2n**
3. **A3: f3(n) = 50n + 100**

Bestimmen Sie, welcher Algorithmus am schnellsten ist, abhängig von der Größe von n. Vergleichen Sie die Algorithmen paarweise und lösen Sie die entsprechenden Ungleichungen, um die Schnittpunkte zu finden.

Aufgabe 4:

Schreiben Sie einen Algorithmus zur Berechnung des größten gemeinsamen Teilers (GGT) von zwei Zahlen m und n mithilfe des Euklidischen Algorithmus. Bestimmen Sie die Zeitkomplexität des Algorithmus in O(f(n))-Notation.

Aufgabe 5:

Angenommen, ein Algorithmus hat eine Zeitkomplexität von O(n^3). Wie viele Operationen würden in etwa benötigt, wenn n:

a) 10 ist
b) 100 ist
c) 1.000 ist