Aufgabe 1)

a) $f = O(n^3)$
Ja, da O-Notation die Laufzeit von Algorithmen nach oben hin abschätzt, es wird also vom worst case ausgegangen. Dabei wird das größte Wachstum in Abhängigkeit der Problemgröße n gesucht. In der vorliegenden Funktion steigen die Kosten beim logarithmischen Wachstum log2(n7) langsam. Die Kosten beim linearen Wachstum 24n sind höher.

b) Ja, denn die Laufzeit bei $O(n^3)$ ist höher als bei O(n2 _ log2(n) + n7/n3)

Aufgabe 2)

S = {1, 4, 2, 7}, c = 7
Zuweisungen: 4
Vergleiche: 8
➔ 12 Elementaroperationen

S = {1, 2, 3, 4, 5, 19, 49, 50}, c = 8
Zuweisungen: 9 Zuweisungen
Vergleiche: 17
➔ 26 Elementaroperationen

Tworst/erfolgreich(n) = 3n
Tworst/erfolglos(n) = 3n + 2

Aufgabe 3d)

Es befinden sich keine geraden Zahlen im Array
Tbest(n) = 3n + 3 = O(n)
Es befinden sich nur gerade Zahlen im Array
Tworst(n) = 4n + 3 = O(n)

Aufgabe 4)

```
public int multiply(int[][] matrix, int[] vector) {

T1 (n) = O (f(n)) = O(1) → da ein Vergleich
if (matrix[0].length != vector.length) {
throw new RuntimeException(
"matrix.length and vector.length must be the same.");
            }

            T2 (n) = O(1) → da eine Zuweisung
            int[] result = new int[matrix.length];

            T3(n) = O(1) → da eine Zuweisung
            int counter = 0;

            T4a(n) = O(f(n)*g(n)) = 0(n)
            for (int i = 0; i < matrix.length; i++) {

                  counter = counter + 2;

                  T4b(n) = O(n)
                  for (int j = 0; j < vector.length; j++) {
                        counter = counter + 2;
                        result[i] += matrix[i][j] * vector[j];
```

```
                    counter++;

                }
            }
```

➔ T4a/b = O(n * n) = O(n²)

➔ Unwichtige multiplikative Therme, die durch Zuweisungen innerhalb der for-Anweisungen entstehen, können ignoriert werden

```
            T5(n) = O(1) → da eine Zuweisung
            resultVector = result;

            return counter;
        }
```

T(n) = O(max(f(n),g(n))) = O(n²)

➔ Die Sequenz mit dem höchsten Aufwand im Algorithmus (also T4a/b(n)) entspricht den Gesamtkosten T(n) = O(n²)