Aufgabe 4

(a) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```java
int mystery(int n) {
  int a = 0, b = 0;
  int i = 0;
  while (i < n) {
    a = b + i;
    b = a;
    i = i + 1;
  }
  return a;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/o_notation/Mystery1.java`

Bestimmen Sie die asymptotische worst-case Laufzeit des Code-Beispiels in $\mathcal{O}$-Notation bezüglich der Problemgröße n. Begründen Sie Ihre Antwort.

Die asymptotische worst-case Laufzeit des Code-Beispiels in $\mathcal{O}$-Notation ist $\mathcal{O}(n)$.

Die `while`-Schleife wird genau n mal ausgeführt. In der Schleife wird die Variable `i` in der Zeile `i = i + 1;` inkrementiert. `i` wird mit 0 initialisiert. Die `while`-Schleife endet, wenn `i` gleich groß ist als `n`.

(b) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```java
int mystery(int n) {
  int r = 0;
  while (n > 0) {
    int y = n;
    int x = n;
    for (int i = 0; i < y; i++) {
      for (int j = 0; j < i; j++) {
        r = r + 1;
      }
      r = r - 1;
    }
    n = n - 1;
  }
  return r;
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/o_notation/Mystery2.java`

Bestimmen Sie für das Code-Beispiel die asymptotische worst-case Laufzeit in $\mathcal{O}$-Notation bezüglich der Problemgröße n. Begründen Sie Ihre Antwort.

`while`: n-mal

1. `for`: $n, n-1, \dots, 2, 1$

2. `for`: $1, 2, \dots, n-1, n$

$n \times n \times n = \mathcal{O}(n^3)$

(c) Bestimmen Sie eine asymptotische Lösung (in Θ-Schreibweise) für die folgende Rekursionsgleichung:

$$T(n) = T\left(\frac{n}{2}\right) + \frac{1}{2}n^2 + n$$

Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a =$ Anzahl der Unterprobleme in der Rekursion

$\frac{1}{b} =$ Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird

$f(n) =$ Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen

Dann gilt:

1. Fall: $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls $f(n) \in \mathcal{O}\left(n^{\log_b a-\varepsilon}\right)$ für $\varepsilon > 0$

2. Fall: $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls $f(n) \in \Theta\left(n^{\log_b a}\right)$

3. Fall: $T(n) \in \Theta(f(n))$

falls $f(n) \in \Omega\left(n^{\log_b a+\varepsilon}\right)$ für $\varepsilon > 0$ und ebenfalls für ein c mit $0 < c < 1$ und alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

1

Anteil Verkleinerung des Problems (b):

um $\frac{1}{2}$ also $b = 2$

Laufzeit der rekursiven Funktion ($f(n)$):

$\frac{1}{2}n^2 + n$

Ergibt folgende Rekursionsgleichung:

$T(n) = 1 \cdot T\left(\frac{n}{2}\right) + \frac{1}{2}n^2 + n$

Nebenrechnung: $\log_b a = \log_2 1 = 0$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a-\varepsilon}\right)$:

$\frac{1}{2}n^2 + n \notin \mathcal{O}\left(n^{-1}\right)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$\frac{1}{2}n^2 + n \notin \Theta(1)$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$:

$\varepsilon = 2$
$\frac{1}{2}n^2 + n \in \Omega\left(n^2\right)$
Für eine Abschätzung suchen wir eine Konstante, damit gilt:
$1 \cdot f(\frac{n}{2}) \leq c \cdot f(n)$
$\frac{1}{2} \cdot \frac{1}{4}n^2 + \frac{1}{2}n \leq c \cdot (\frac{1}{2} \cdot n^2 + n)$
Damit folgt $c = \frac{1}{4}$
und $0 < c < 1$

$\Rightarrow \Theta(\frac{1}{2}n^2 + n)$

$\Rightarrow \Theta(n^2)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha