

# Übung 10 Lösung

## Aufgabe 2

- (a) Die Funktion überprüft, ob das Produkt zweier Vektorelemente gleich der Summe aller Elemente ist:

Die erste *for-Schleife* addiert alle Elemente des Vektors, daher hat dieser Abschnitt eine Laufzeit von  $\mathbf{n}$ . Als nächstes wird erneut der Vektor durchlaufen, und dies zwei Mal (ersichtlich an den Iteratorvariablen  $i$  und  $j$ ). Innerhalb dieser 2 Schleifen wird nun geschaut, ob das Produkt der zwei Einträge  $v[i]$  und  $v[j]$  gleich der in der vorherigen Schleife errechneten Summe ist. Wurden diese 2 Einträge gefunden, so brechen die 2 Schleifen ab und die Funktion gibt *true* zurück, andernfalls haben die zwei Schleifen eine Laufzeit von  $\mathbf{n^2}$ .

Dieser Algorithmus terminiert also spätestens nach  $\mathbf{n^2 + n}$  Schleifendurchläufen und hat damit eine Laufzeit von  $n^2 + n \in O(n^2)$ , jedoch ist der Algorithmus nicht korrekt, da in den beiden Schleifen nicht geprüft wird, ob  $v[i] == v[j]$  ist, sprich, ob ein Vektorelement mit sich selbst multipliziert wird.

- (b) Die Funktion addiert die Elemente an geraden (even) Index-Positionen in `sum_even_odd[0]` und die Elemente an ungeraden (odd) Index-Positionen in `sum_even_odd[1]`:

Zu Beginn wird ein Vektor mit 2 Einträgen, welche mit 0 initialisiert werden, erstellt. Für die Bestimmung der Laufzeit ist dieser Befehl uninteressant. Als nächstes folgen 2 *for-Schleifen*, wobei der Iterator **i** der äußeren Schleife die Werte 0 und 1 annimmt. In der inneren Schleife wird der Iterator **j** immer um 2 erhöht, was zur Folge hat, dass im 1. äußeren Schleifendurchlauf zunächst alle geraden Einträge durch  $v[i + j]$  (also  $v[0 + j]$ ) addiert werden ( $i = 0$ ) und im 2. Durchlauf ( $i = 1$ ) alle ungeraden Einträge ( $v[1 + j]$ ) miteinander addiert werden.

Der Algorithmus funktioniert gemäß seiner Beschreibung und hat eine Laufzeit (terminiert spätestens in) von  $2 \cdot \frac{n}{2} \in O(n)$ .

- (c) Die Funktion hängt für jedes Element im Vektor die Zahl mit umgedrehten Vorzeichen an:

Interessant an diesem Algorithmus ist, dass die Abbruchbedingung das Ende des Vektors ist, diese aber nicht vorher in einer separaten Variable festgehalten wird, sondern sich nach jedem Schleifendurchlauf neu berechnet, was zur Folge hat, dass dieser Algorithmus nicht terminieren kann, da mit jedem Schleifendurchlauf der Vektor größer wird und das Ende nie erreicht werden kann.

- (d) Die Funktion berechnet die Summe der Elemente an den Index-Positionen, die eine Zweierpotenz sind:  $2^0, 2^1, 2^2, 2^3, \dots$  (sofern dies ein gültiger Index, also kleiner als die Länge des Vektors ist)