

Aufgabe 1. Durch iteratives Einsetzen:

$$T(0) = 0$$

$$T(1) = T(0) + 1^2 = 1$$

$$T(2) = T(1) + 2^2 = 1 + 4 = 5$$

$$T(3) = T(2) + 3^2 = 5 + 9 = 14$$

$$T(4) = T(3) + 4^2 = 14 + 16 = 30$$

Dies lässt sich für jede natürliche Zahl bis n fortsetzen, wodurch man die allgemeine Formel $T(n) = 1^2 + 2^2 + 3^2 + \dots + n^2$ für die Rekursionsgleichung erhält. Mathematisch gesehen ist dies:

$$\sum_{i=0}^n i^2 = \frac{1}{6} \cdot (n \cdot (n+1) \cdot (2n+1)).$$
 Da diese Summenformel ein

Polynom 3. Grades ist, kann folgende Abschätzung angegeben werden:

$$T(n) = O(n^3)$$

Aufgabe 2.

Laufzeit der naiven Strategie: $O(n^2)$

Diese Laufzeit ergibt sich aus dem Kopiervorgang von

0 bis $n-1$ Elementen in ein ^{jeweils} neues Array. Nach der

Gaußschen Summenformel ergibt sich: $\sum_{i=0}^{n-1} i = \frac{n \cdot (n-1)}{2}$. (Summe der Kosten der for-Schleifen fürs Kopieren)

Somit ein Polynom 2. Grades und daher $O(n^2)$

Laufzeit der Doubling-Strategie: $O(\log n)$

Da nicht immer ein neues Array erstellt werden muss, muss auch nicht jedes mal das gesamte Array kopiert werden.


Zwischendrin werden lediglich Elemente in das bestehende Array eingefügt,

da es noch groß genug ^{immer} ist. Deshalb werden die Elemente

des Arrays auch nicht bei neuen Elementen, die hinzukommen,

mitkopiert. ~~Bei n Elementen~~ Bei n Elementen

~~ergeben sich~~ ^{ergeben} sich somit $\log_2 n$ Kopiervorgänge, da bei größeren n immer

BRUNNEN  weniger kopiert und neu eingefügt wird. Daher für das

Kopieren $O(\log_2 n)$

amortisierten Kosten: $O(1)$, da das Einfügen ~~des~~ Laufzeitkosten von $O(n)$ hat, die bei dem Kopieren des Elements in ein neues Array entstehen. Wenn man nun die durchschnittskosten betrachtet, erhält man $O(\frac{n}{n}) = O(1)$.

best-case Kosten: $O(1)$. Dies tritt auf, wenn ein Array bereits groß genug ist und nur ein weiteres Element ins Array gespeichert werden muss. Somit konstant und $O(1)$.

worst-case Kosten: $O(n)$. Dies tritt auf, wenn ein Array voll ist und verdoppelt werden muss. Die Arrayelemente müssen kopiert werden und das neue Element eingefügt werden, also $O(n+1) = O(n)$ (for-Schleife).

Aufgabe 4 a) ~~[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]~~

Index start: 0

Vertauschungen: (0,4); (1,3); (1,4); (2,3); (2,4); (3,4)