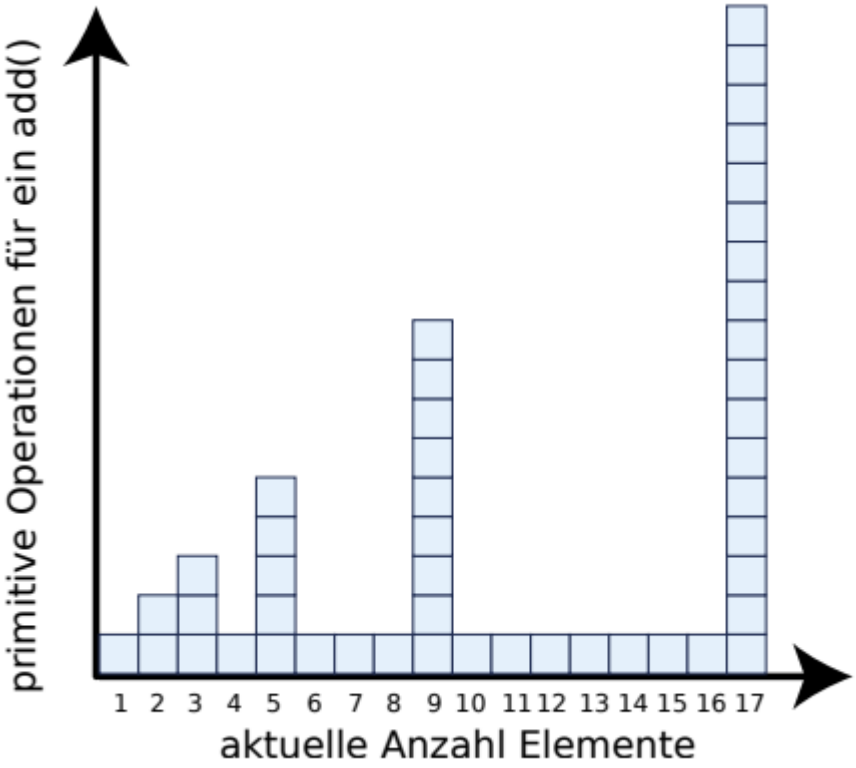
Blatt 4

Aufgabe 1



Bei einem $add()$ Aufruf erfolgt die Verdopplung immer $2^i + 1$, wobei wir bei 1 starten. Das $add()$ dauert somit $2^i + 1$ Zeiteinheiten. Wenn wir das Analog zum $remove()$ betrachten, dann könnten wir sagen das die Halbierung immer bei $2^i - 1$ Zeiteinheiten passiert. Wenn wir nun anehmen das eine Zeiteinheit 1€ kostet, dann schneiden wir immer ausgehend davon die Hälfte vom Turm ab und kippen es nach links. So passt immer der Turm rein und wir haben im gesamten die Kosten von 2€. betrachten wir den schlechtesten Fall, heißt wenn $add()$ und $remove()$ hintereinander ablaufen. Da würde es Sinn machen immer um $\frac{size()}{4}$ zu removen und wenn ein Viertel erreicht wurde, es um $\frac{1}{2}$ zu removen.

Aufgabe 2

1. L
2. L = [10]
3. L = [10, 1]
4. L = [5,10,1] → [5, 8, 10 , 1]
5. L = [5, 2, 8, 10, 1]
6. L = L.addBefore(3,9) → [5, 2, 8, 9, 10, 1]
7. L = L.remove(L.before(L.before(2))) → L.remove(1) → [2, 8, 9, 10, 1]

Aufgabe 3

1. Bei einer verketteten Liste gibt es keine indizes, da diese ja immer größer werden können oder kleiner und darum nicht fix sind. Darum ist eine $for()$-Loop nicht angemessen, da dieser über die indizes iteriert. Bei einem Iterator kann immer vor und zurück bewegt werden und es hat die Möglichkeit abzufragen ob mehr Elemente dahinter stehen oder nicht. Der Sinn eines iterators ist es keinen Bezug auf die Datenstruktur zu nehmen.

   $iterator = \mathcal{O}(n), for - loop = \mathcal{O}(n^2)$
2. Ein Schnappschuss-Iterator erstellt eine lokale Kopie der Daten(z.B Liste) und geht dann über die Elemente drüber. An der Laufzeitkomplexität ändert sich nichts da das zuweisen einer Datenstruktur in eine lokale Kopie keinen Unterschied macht.
3. Man sollte beachten wo genau man startet, denn wenn man von Anfang an die Elemente removed, dann ist das nicht mehr der Sinn der dynamischen Verkleinerung. Mit einem Iterator kann man ja auch vom letzten Element aus iterieren starten und so kann man dieses Problem umgehen.

Aufgabe 4

Leider nicht vollständig geschafft, darum nicht abgegeben.