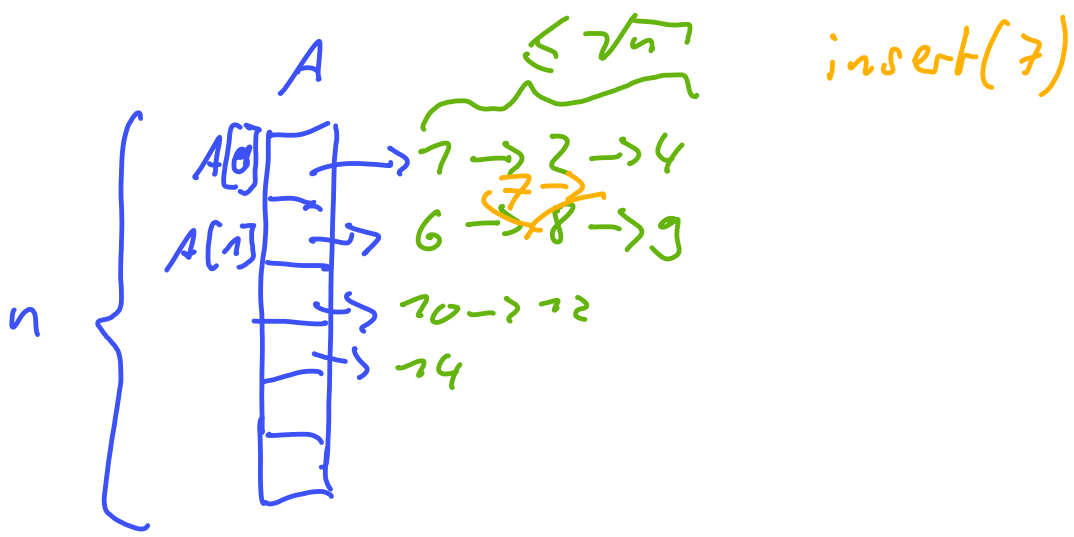
Operation find:

- Binäre Suche um richtige Liste zu finden $\Theta(\log n)$
- lineare Suche in der Liste $\Theta(\sqrt{n})$



Operation insert:

- suche wie bei find die richtige Position zum Einfügen $\Theta(\sqrt{n})$
- füge an entsprechender Stelle ein
- Falls Liste in die eingefügt wurde $> \sqrt{n}$ Elemente enthält:
  - ↳ Komplett neues Array mit Liste der Länge 1 pro Eintrag
  - ↳ neues n heißt n' $\Theta(n')$

$\Rightarrow$ Worst case: $\Theta(n')$ pro Operation

Amortisierte Sichtweise:

- Viele Operationen günstig: $\Theta(\sqrt{n})$
- vor jeder teuren Operation: viele günstige Operationen.

Charging Methode:

- teure Operation: Kosten n'
- Es gibt Listen, die $> \sqrt{n}$ viele Elemente enthält.
- Verteile n' Kosten auf die $\sqrt{n}$ Einfügeoperationen dieser $\sqrt{n}$ Elemente
- Einfügeoperation von vorher bekommt $\frac{n'}{\sqrt{n}}$ viele Kosten
- Auf jede Operation wird nur einmal geschaut $\rightsquigarrow \Theta\left(\frac{n'}{\sqrt{n}}\right)$

Kontomethode:

- Günstige Einfügeoperation: Zahle zusätzlich n ein $\rightsquigarrow \Theta(n)$
- teure Operation: hebe n' ab $\rightsquigarrow$ amortisiert kostenlos
- Zeige: Konto wird nie negativ.

  $\hookrightarrow n' \leq n \cdot \sqrt{n}$

  $\hookrightarrow$ Wenn es Liste der Länge $k+1$ gibt, dann ist der Kontostand $\geq k \cdot n$

  $\hookrightarrow$ teure Operation $\Rightarrow \exists$ Liste der Länge $\sqrt{n}+1 \Rightarrow$ Kontostand $\geq \sqrt{n} \cdot n$

Potential: $\Phi(A) = k \cdot n$ wobei k Länge der längsten Liste ist

- A' sei die DS nach der Operation.
- amort. Kosten = tatsächliche Kosten + $\Phi(A') - \Phi(A)$
- $\Phi(A) \geq 0$