Motivation

- seltene schlechte Laufzeiten mit vielen effizienten Laufzeiten
    - nicht automatisch ineffizient
- Begründung
    - einzelne Operationen sind aufwendig
    - m aufeinander folgende Operationen sind dennoch effizient
    - seltene schlechte Laufzeit wird auf häufige gute Laufzeit aufgeteilt

Beispiel [[Dynamische Arrays]]

- n-mal Einfügen und einmal erweitern in $O(n)$
    - $n * O(1) + O(n) = O(2n) = O(n)$

**Beispiel Dynamisches Array**

- add/delete hat im WC Laufzeit $\Omega(n)$.
- Wir betrachten $k$ add/delete-Operationen auf einem Anfangs leerem Array.
- Benötigen wir dann $\Omega(k^2)$ Laufzeit?

**Nein:**

Man betrachte $k$ aufeinanderfolgende `add` oder `delete`-Operationen in beliebiger Reihenfolge auf einem anfangs leeren dynamischen Array.
Dann ist die Laufzeit für diese $k$ Operationen $T(k) = O(k)$.

- Man betrachte $k$ aufeinanderfolgende `add` oder `delete`-Operationen in beliebiger Reihenfolge auf einem anfangs leeren dynamischen Array.
Dann ist die Laufzeit für diese $k$ Operationen $T(k) = O(k)$.

| i-te Operation | Auszahlung $a_i$ (Laufzeit) | Einzahlung $e_i$ |
| --- | --- | --- |
| add (ohne Umstrukturierung) | 1 | 3 |
| add (mit Umstrukturierung) | $n_i$ | 2 |
| delete (ohne Umstrukturierung) | 1 | 3 |
| delete (mit Umstrukturierung) | $n_i$ | 2 |

$n_i$ …… … *Anzahl der Elemente in Array nach Operation i*
$n_{cap,i}$ … *Größe des Arrays nach Operation i*
$K_i$ … …… *Kontostand nach Operation i,* $K_0 = 0$

- 