Eigenschaften

- worst-case optimal
- adaptiv

  Maß für Sortiertheit: #Fehlstände

  – sei $a_1, \ldots, a_n$ eine (unsortierte) Zahlenfolge

  – f_i = # Fehlstände für a_i

  $f_i = |\{a_j | j > i, a_j < a_i\}|$

  – Fehlstände der Zahlenfolge: $F = \sum_{i=1}^{n} f_i$

  – Es gilt: $0 \leq F \leq \frac{n(n-1)}{2} = \binom{n}{2}$

Motivation

- Grundidee
  - Einfügen in verkehrter Reihenfolge
  - innere Knoten speichern Maximum des Teilbaums
- Baumsortieren
  - allgemein beim Einfügen von a_i

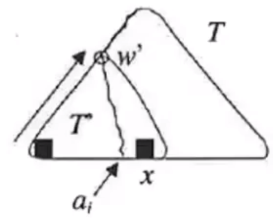
    $a_n, \ldots, a_{i+1}$ sei bereits eingefügt (sind sortiert im Baum)

    * a_i wird jetzt eingefügt

      Folge: $a_1 \ldots a_i \quad \underbrace{a_{i+1} \ldots a_n}_{\substack{bereits\ eingefügt, \\ davon\ f_i\ Zahlen < a_i}}$

      Es folgt:

    * a_i wird im Baum an der Stelle $f_i + 1$ von links eingefügt.



w'…Wurzel von T'

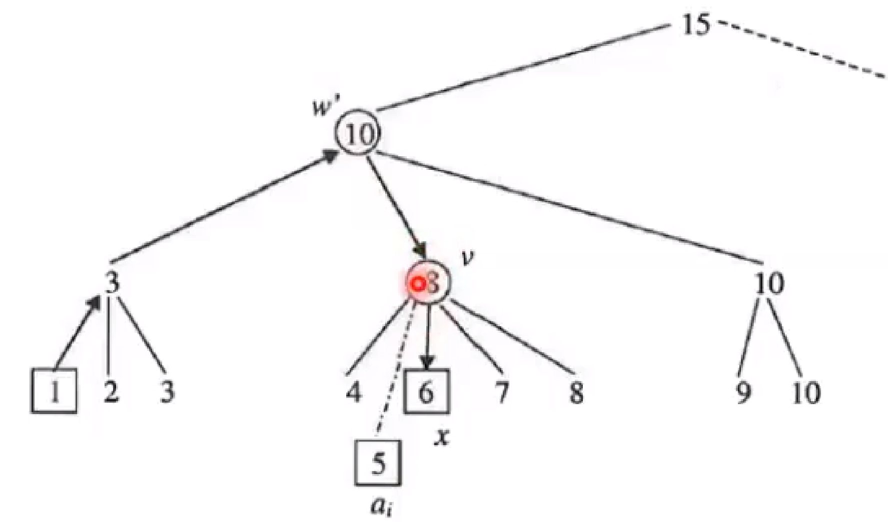
Idee: Wenn f_i klein, liegt
w' tief.

Einfügen bottom up:

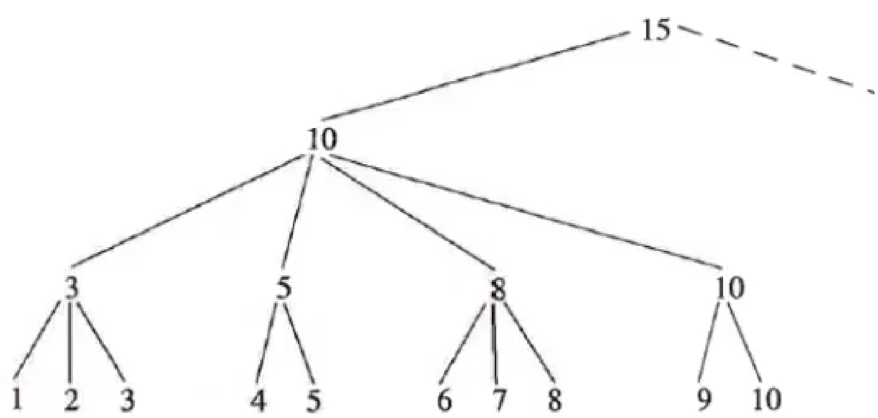
– Starte bei linkestem Blatt
– Laufe bis zu w', d.i. erster innerer Knoten mit Max$\geq a_i$.
– Füge a_i in Teilbaum ein.

→*Anhängen* von a_i in $\theta(\log f_i)$ Zeit.

a_i=5 eingefügt



SPALTE(v) liefert:



Laufzeit B-Sort:

$$T(n) = O\left(n \log\left(\frac{F}{n}\right) + n\right)$$

F…Anzahl der Fehlstände in Zahlenfolge

- F=O(n) → T(n) = O(n)
- F=O(n^2) → T(n) = O(n log n)