

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 8: LISTEN, BÄUME & SORTIEREN

Eric Kunze

`eric.kunze@mailbox.tu-dresden.de`

TU Dresden, 12.12.2019

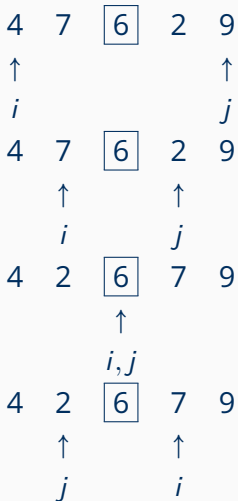
- ▶ C.A.R. HOARE, 1962
- ▶ Idee: ***Divide-and-Conquer***
- ▶ Teilung des zu sortierenden Feldes
- ▶ Ziel: alle Elemente
 - ▷ links des Teilungselements: kleinere Schlüssel
 - ▷ rechts des Teilungselements: größere Schlüssel
- ▶ dazu: Austausch über möglichst große Strecken

QUICKSORT

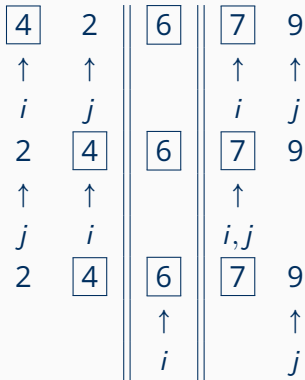
```
1 void quicksort(int a[], int L, int R) {
2     int i, j, w, x, k;
3
4     i = L;  j = R;  k = (L + R) / 2;
5     x = a[k];
6
7     do {
8         while (a[i] < x) i = i + 1;
9         while (a[j] > x) j = j - 1;
10        if (i <= j) {
11            w = a[i];    //
12            a[i] = a[j]; // swap a[i] and a[j]
13            a[j] = w;    //
14            i = i + 1;  j = j - 1;
15        }
16    } while (i <= j);
17
18    if (L < j) quicksort(a, L, j);
19    if (R > i) quicksort(a, i, R);
20 }
```

AUFGABE 3

1. Durchlauf



2. Durchlauf



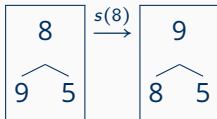
► Heap-Eigenschaft:

- Für jeden Knoten n gilt: Wenn n mit h beschriftet ist, dann müssen die Beschriftungen der Nachfolger von n kleiner als h sein (an der Wurzel eines Teilbaums stets stets der größte Schlüssel des Teilbaums) → *Heap-Eigenschaft*

► Herstellung der Heap-Eigenschaft: Funktion *sinkenlassen*

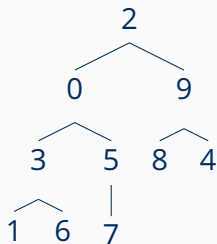
- Sinkenlassen eines Knotens in Richtung Blätter
- Vertauschen von Wurzelbeschriftung mit größerem Kind

► Beispiel:

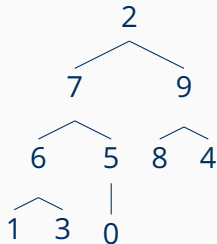


- ▶ Bäume sind nur Veranschaulichung
- ▶ Algorithmus arbeitet auf Listen
- ▶ zwei Phasen
 - ▷ **1. Phase:** Einsortieren in den Heap und Herstellen der Heap-Eigenschaft
 - ▷ **2. Phase:** Führe Sortierschritt wiederholt durch:
 - Tausch von Wurzel und "letztem" Element (tiefste Ebene, ganz rechts)
 - Fixiere dieses Element
 - Sinkenlassen des neuen Wurzelements

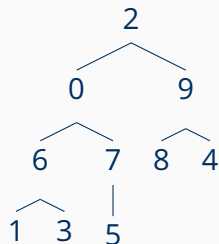
AUFGABE 4 — PHASE 1



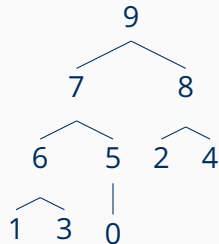
$s(0), s(9)$
→



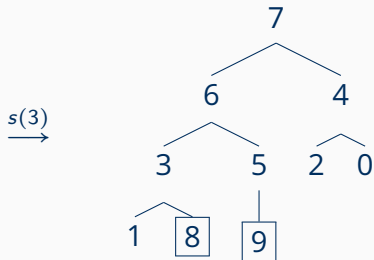
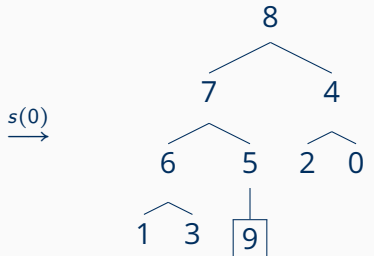
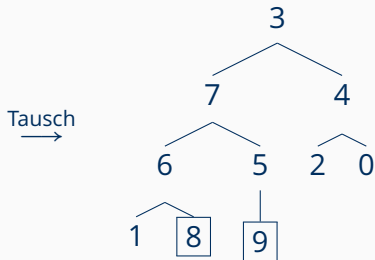
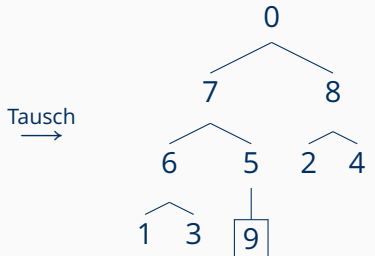
$s(3), s(5)$
→



$s(2)$
→

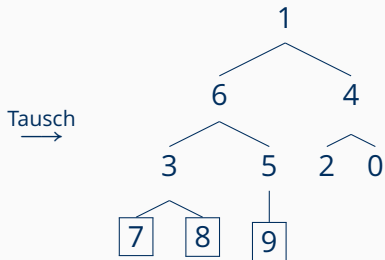


AUFGABE 4 — PHASE 2



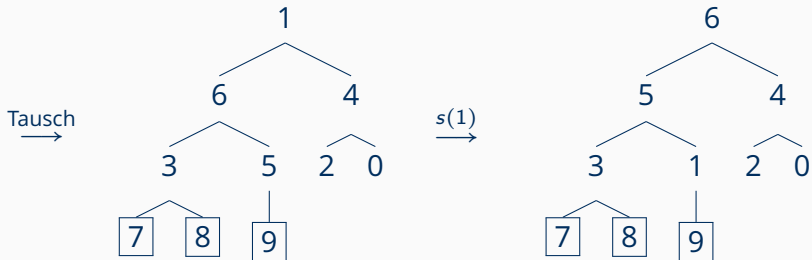
AUFGABE 4 — PHASE 2

Fortsetzung zur Vollständigkeit



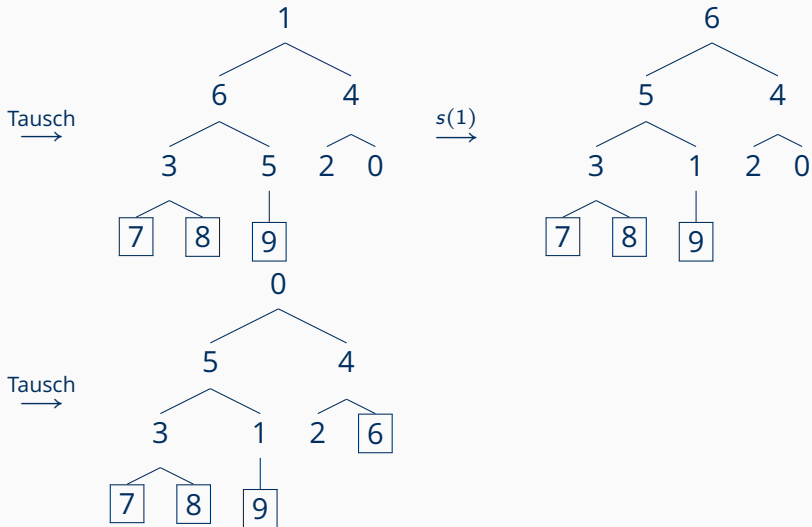
AUFGABE 4 — PHASE 2

Fortsetzung zur Vollständigkeit



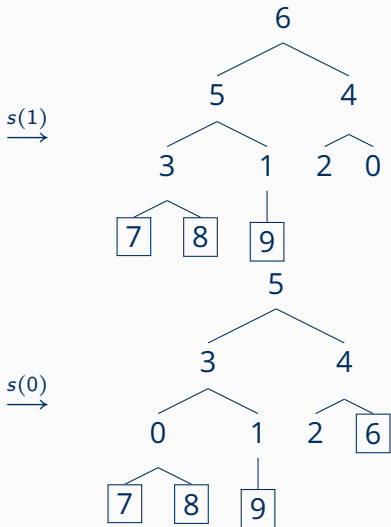
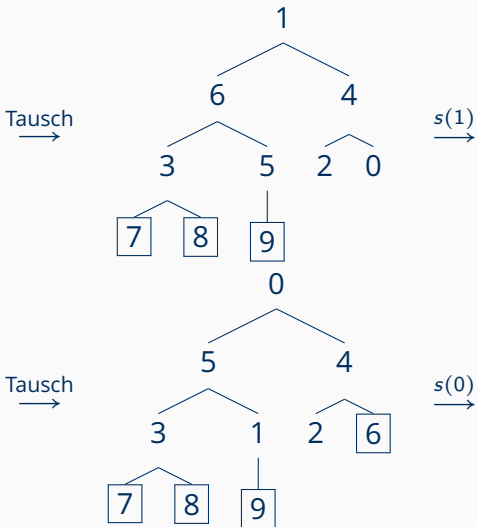
AUFGABE 4 — PHASE 2

Fortsetzung zur Vollständigkeit



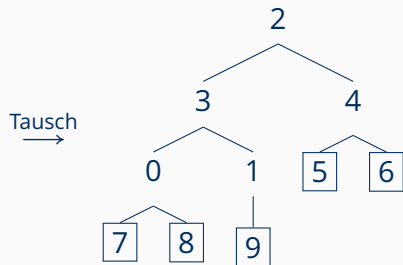
AUFGABE 4 — PHASE 2

Fortsetzung zur Vollständigkeit



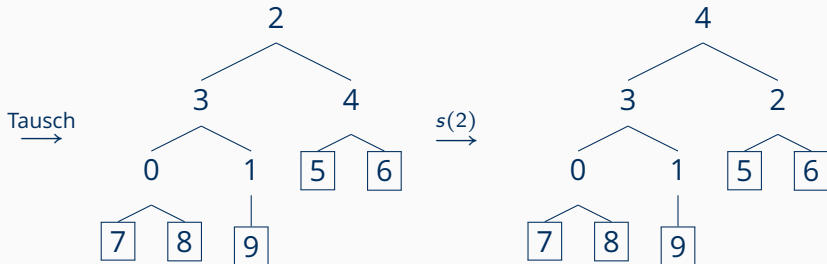
AUFGABE 4

Fortsetzung zur Vollständigkeit



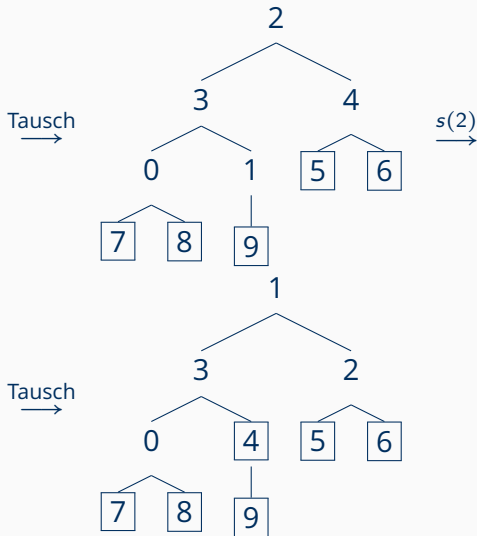
AUFGABE 4

Fortsetzung zur Vollständigkeit



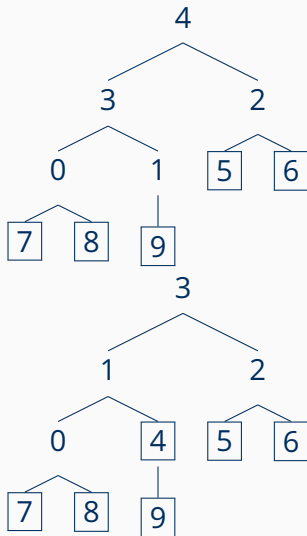
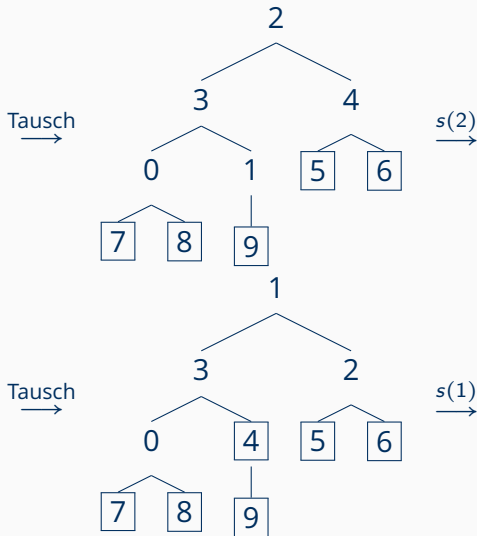
AUFGABE 4

Fortsetzung zur Vollständigkeit



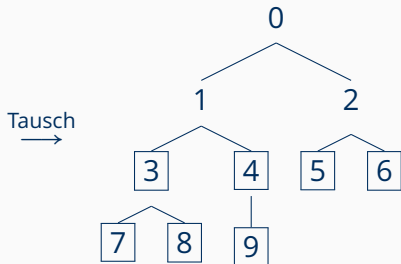
AUFGABE 4

Fortsetzung zur Vollständigkeit



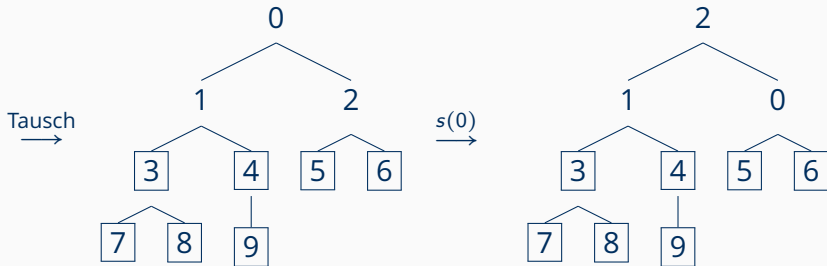
AUFGABE 4

Fortsetzung zur Vollständigkeit



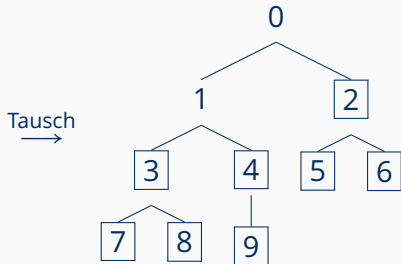
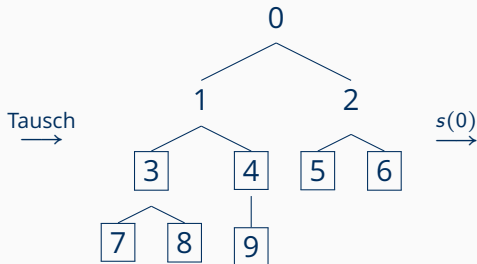
AUFGABE 4

Fortsetzung zur Vollständigkeit



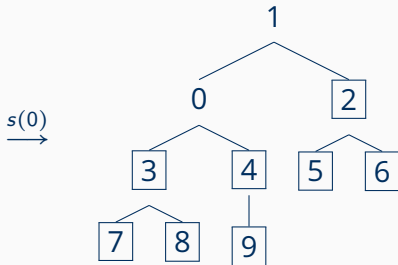
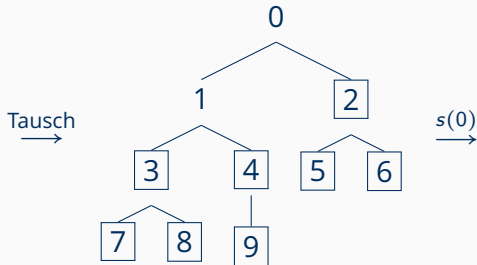
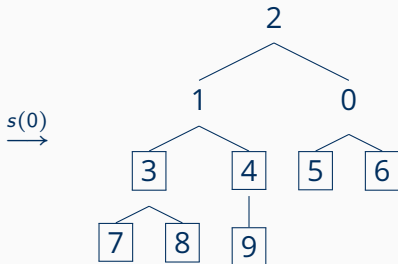
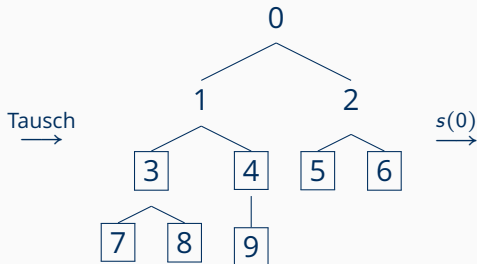
AUFGABE 4

Fortsetzung zur Vollständigkeit



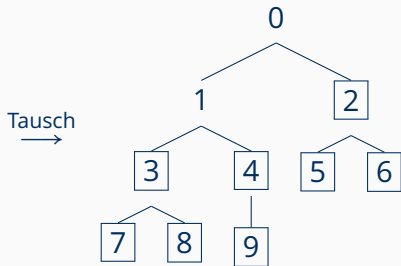
AUFGABE 4

Fortsetzung zur Vollständigkeit



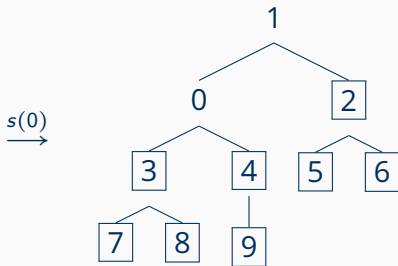
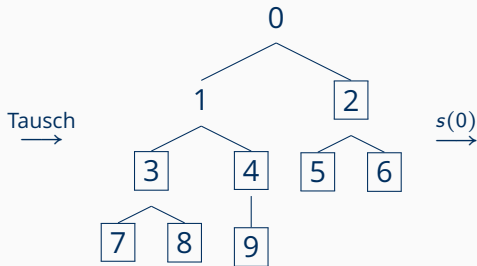
AUFGABE 4

Fortsetzung zur Vollständigkeit



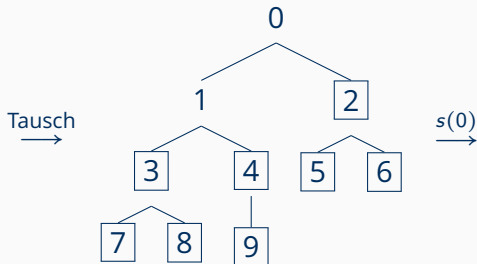
AUFGABE 4

Fortsetzung zur Vollständigkeit

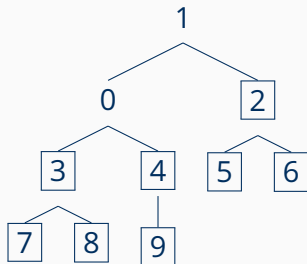


AUFGABE 4

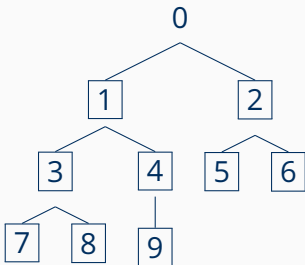
Fortsetzung zur Vollständigkeit



$s(0)$
→

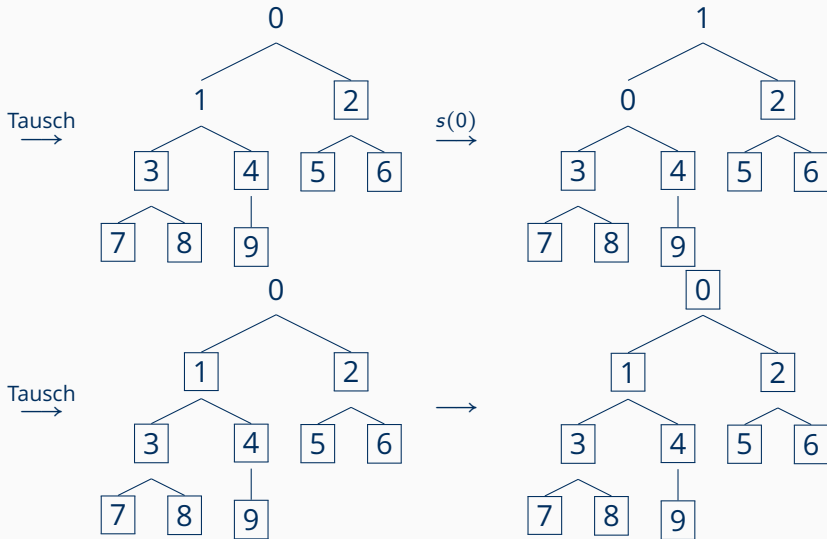


Tausch
→



AUFGABE 4

Fortsetzung zur Vollständigkeit



sortierte Liste:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

FERTIG 😊