TECHNISCHE UNIVERSITÄT DRESDEN

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 7: SORTIEREN

Eric Kunze

eric.kunze@tu-dresden.de

TU Dresden, 24. November 2021

letzte Änderung:
24.11.2021, 21:21

QUICKSORT

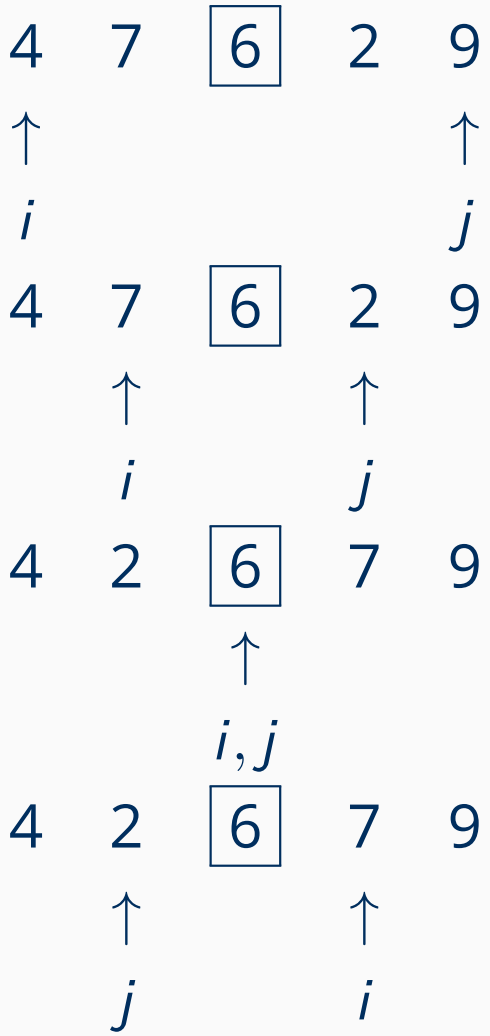
- C.A.R. Hoare, 1962
- Idee: ***Divide-and-Conquer***
- Teilung des zu sortierenden Feldes
- Ziel: alle Elemente
  - links des Teilungselements: kleinere Schlüssel
  - rechts des Teilungselements: größere Schlüssel
- dazu: Austausch über möglichst große Strecken

QUICKSORT

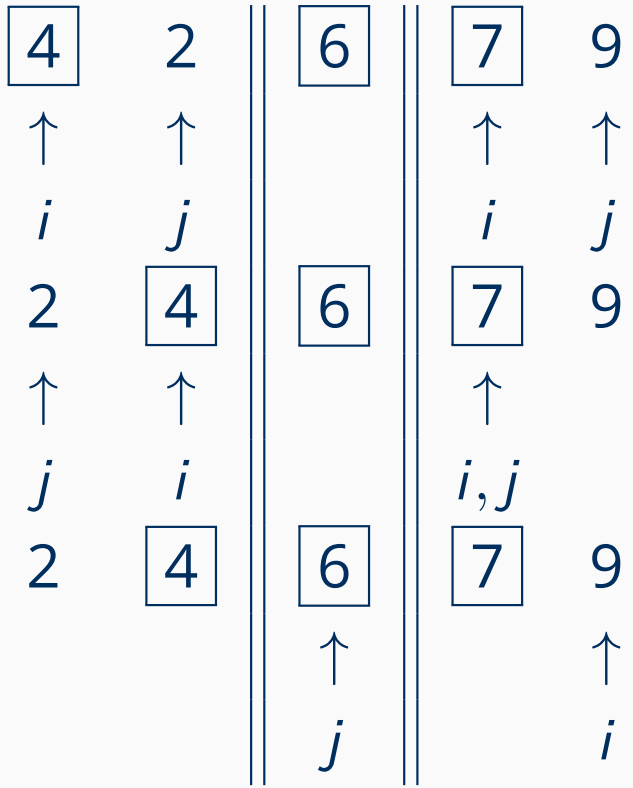
```
1  void quicksort(int a[], int L, int R) {
2      int i, j, w, x, k;
3
4      i = L;  j = R;  k = (L + R) / 2;
5      x = a[k];
6
7      do {
8          while (a[i] < x) i = i + 1;
9          while (a[j] > x) j = j - 1;
10         if (i <= j) {
11             w = a[i];      //
12             a[i] = 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 1

1. Durchlauf

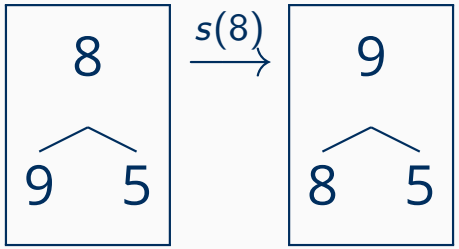


2. Durchlauf



BINÄRBÄUME MIT HEAP-EIGENSCHAFT

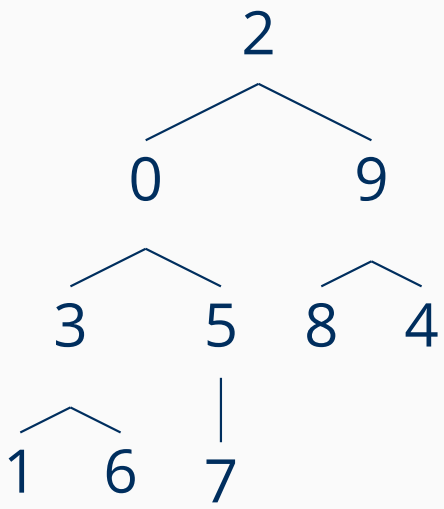
- **Heap-Eigenschaft:**
  - Für jeden Knoten n gilt: Wenn n mit h beschriftet ist, dann müssen die Beschriftungen der Nachfolger von n kleiner als h sein (an der Wurzel eines Teilbaums steht stets der größte Schlüssel des Teilbaums) $\rightarrow$ *Heap-Eigenschaft*
- Herstellung der Heap-Eigenschaft: Funktion *sinkenlassen*
  - Sinkenlassen eines Knotens in Richtung Blätter
  - Vertauschen von Wurzelbeschriftung mit größerem Kind
  - Beispiel:



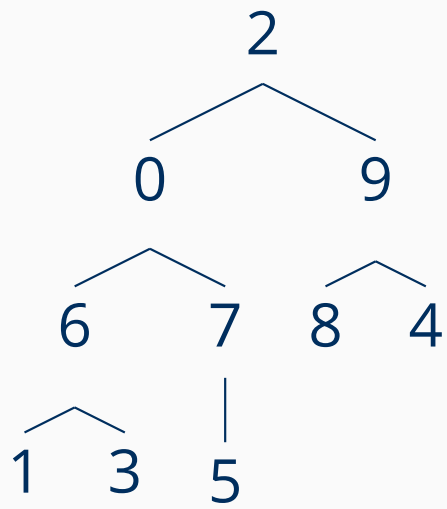
HEAPSORT

- 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 Wurzelelements

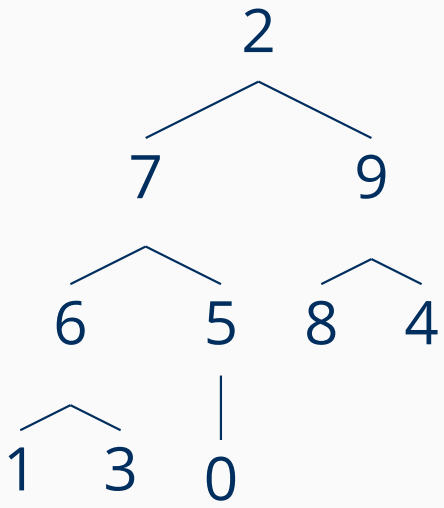
AUFGABE 2 — PHASE 1



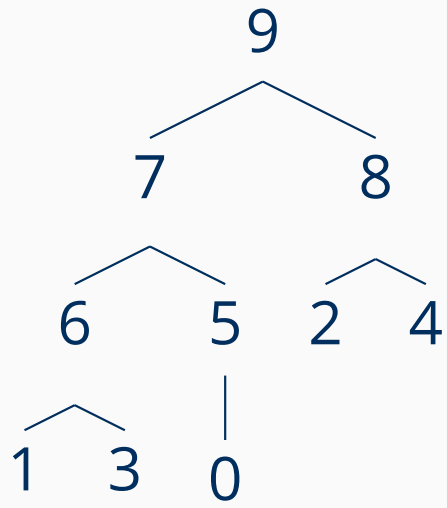
$s(3), s(5)$ →



$s(0), s(9)$ →

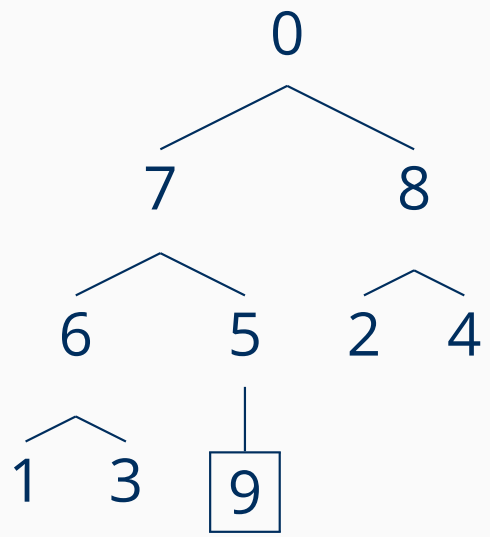


$s(2)$ →

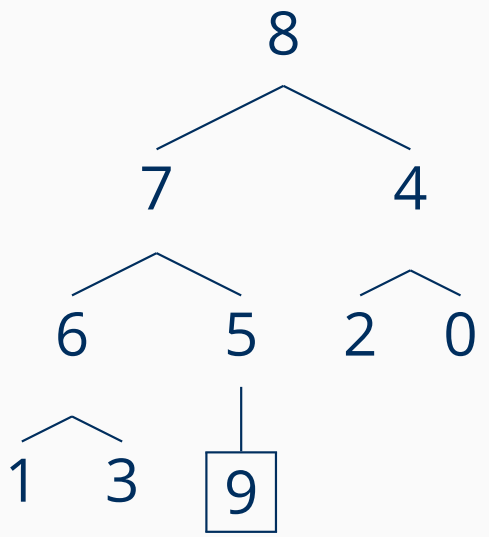


AUFGABE 2 — PHASE 2

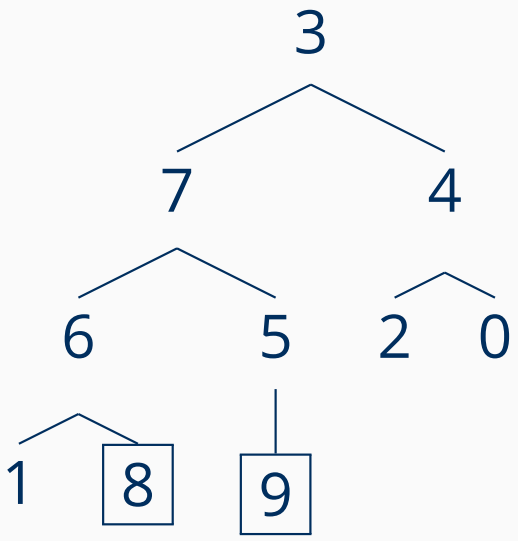
Tausch →



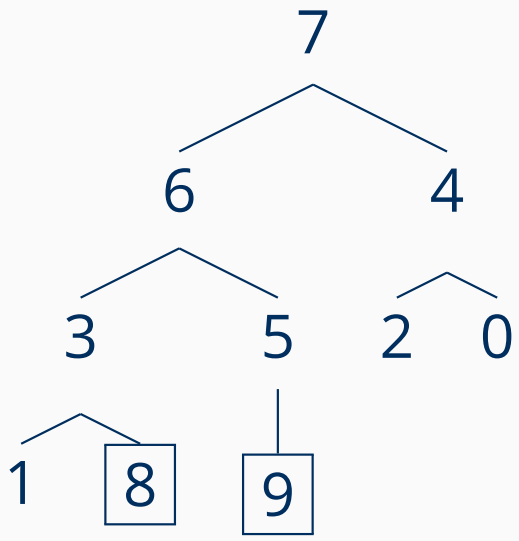
$s(0)$ →



Tausch →



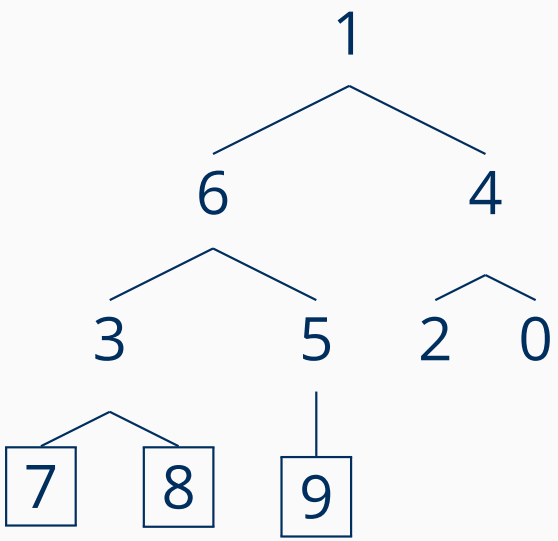
$s(3)$ →



AUFGABE 2 — PHASE 2

Fortsetzung zur Vollständigkeit

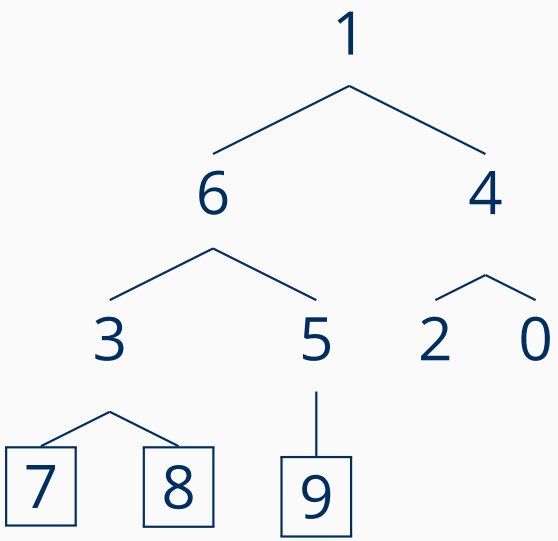
Tausch $\longrightarrow$



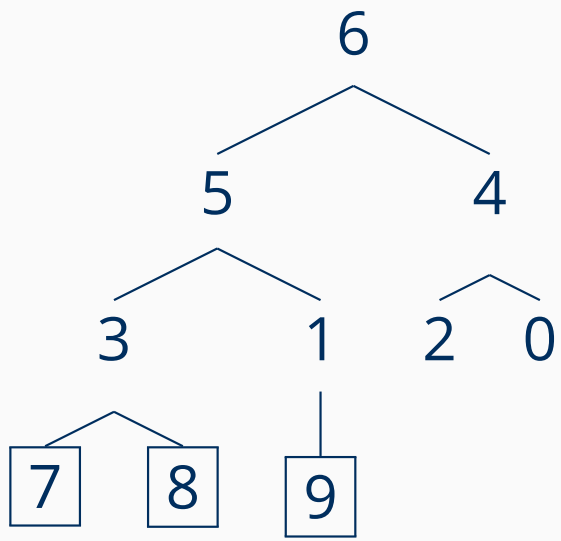
AUFGABE 2 — PHASE 2

Fortsetzung zur Vollständigkeit

Tausch $\longrightarrow$



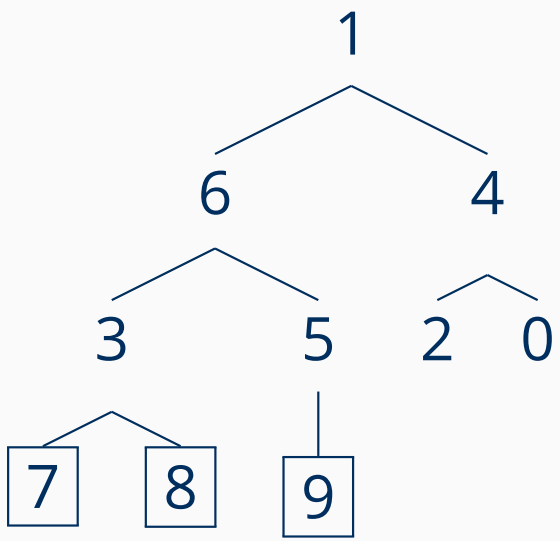
$s(1)$ $\longrightarrow$



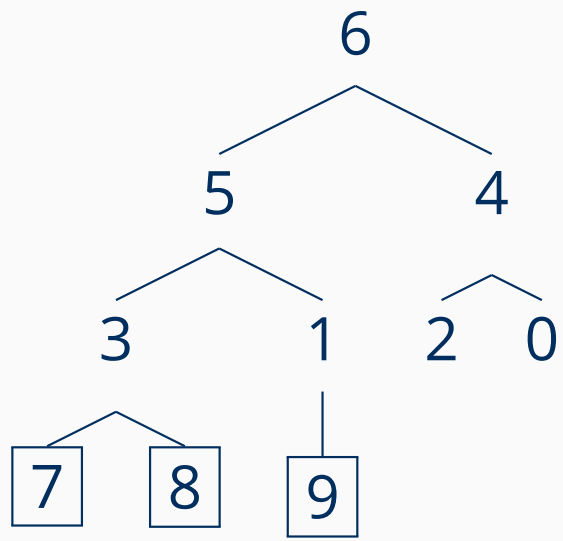
AUFGABE 2 — PHASE 2

Fortsetzung zur Vollständigkeit

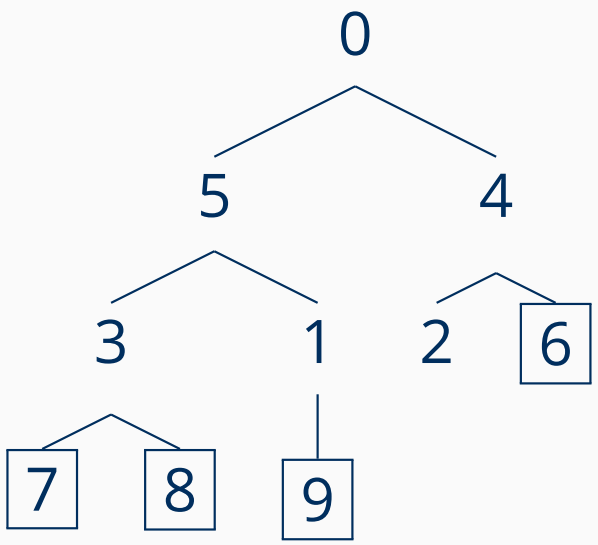
Tausch →



$s(1)$ →

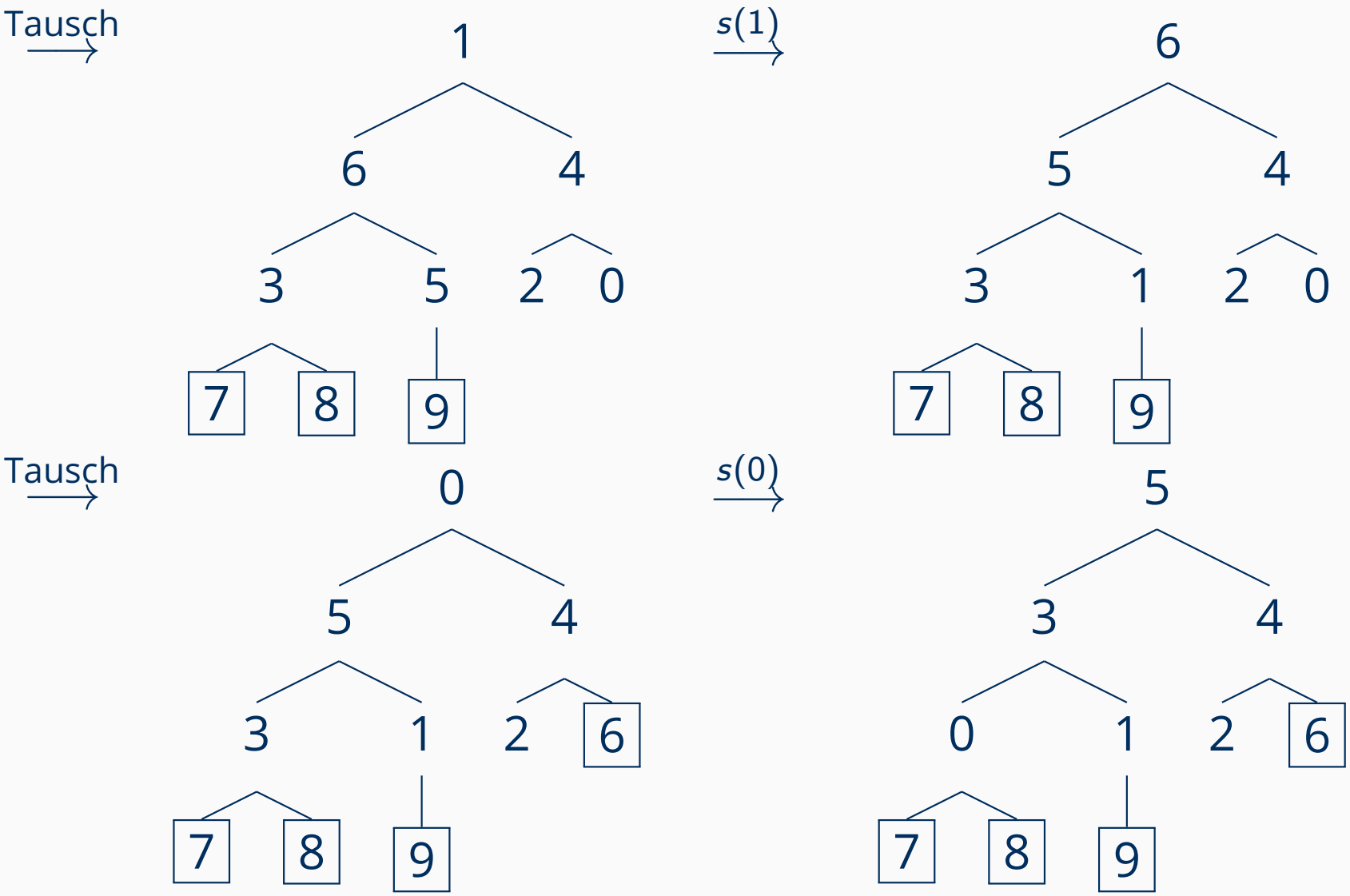


Tausch →



AUFGABE 2 — PHASE 2

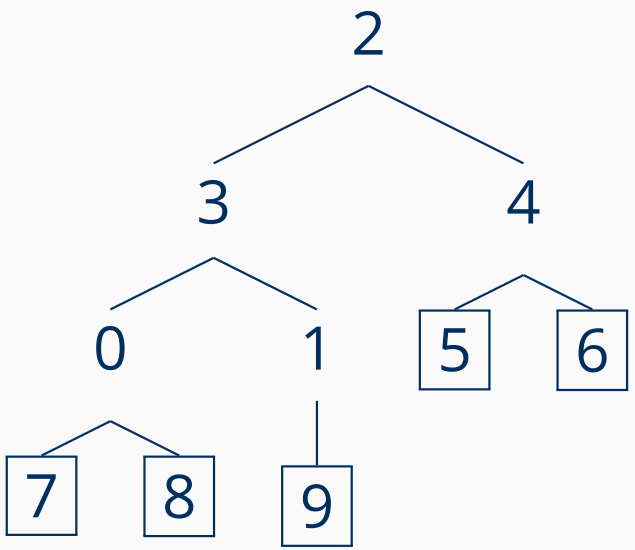
Fortsetzung zur Vollständigkeit



AUFGABE 2

Fortsetzung zur Vollständigkeit

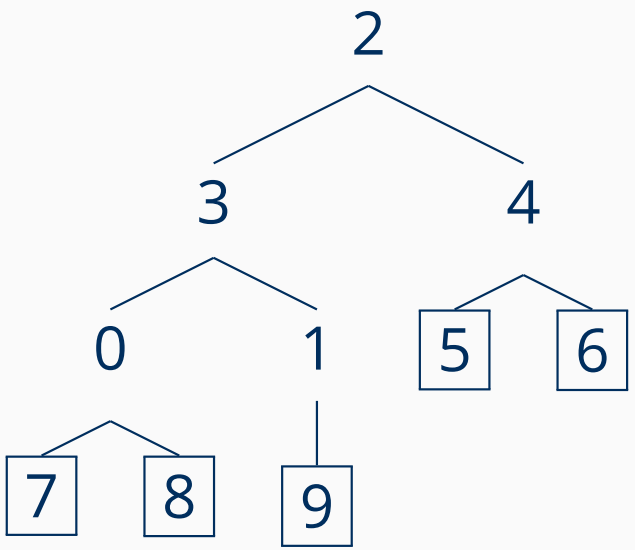
Tausch $\longrightarrow$



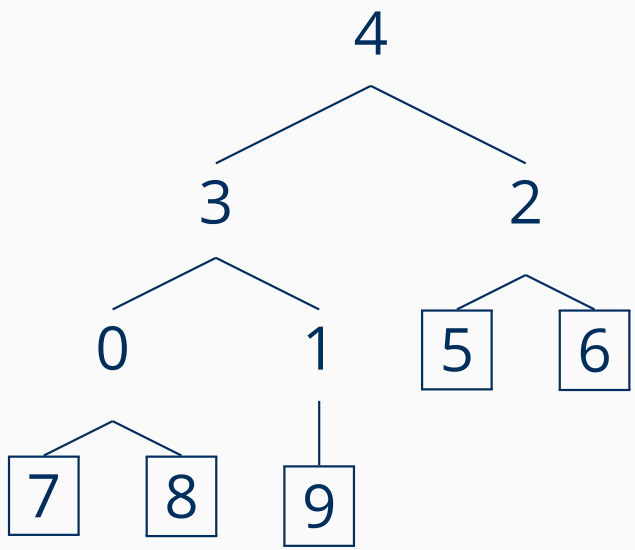
AUFGABE 2

Fortsetzung zur Vollständigkeit

Tausch $\longrightarrow$



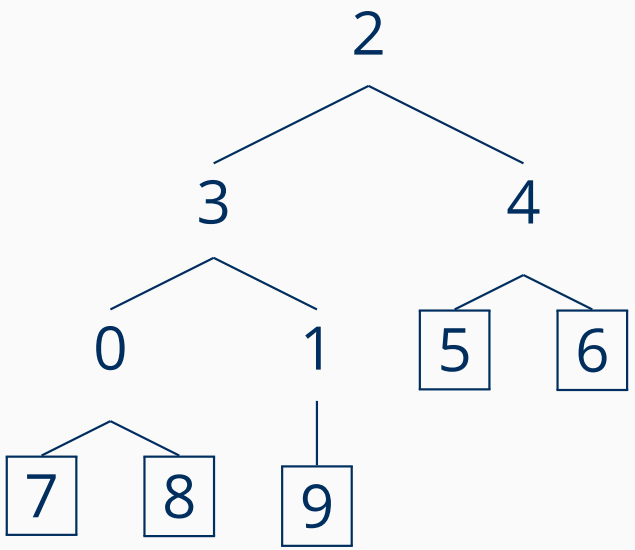
$s(2)$ $\longrightarrow$



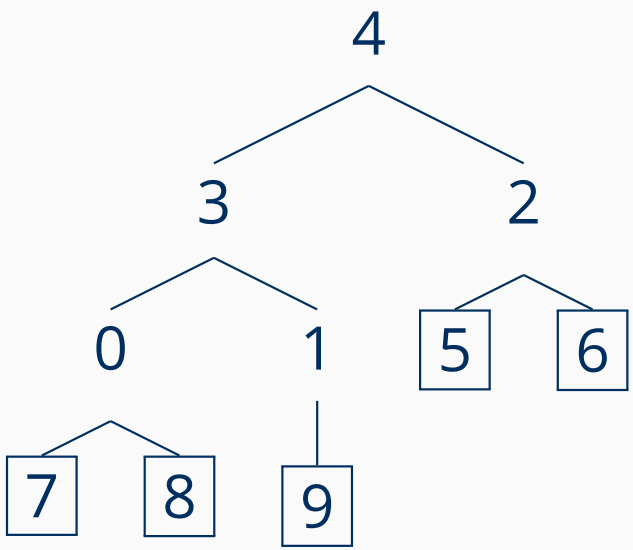
AUFGABE 2

Fortsetzung zur Vollständigkeit

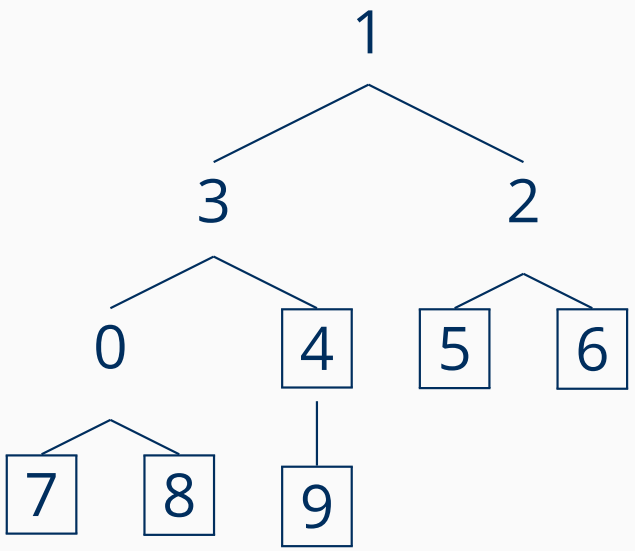
Tausch →



$s(2)$ →



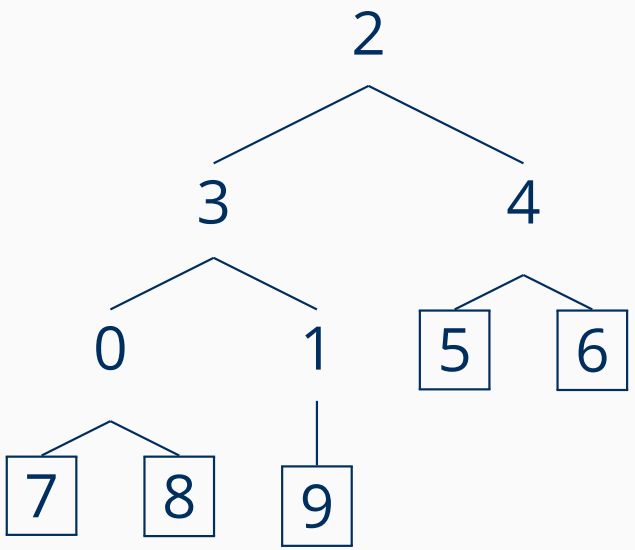
Tausch →



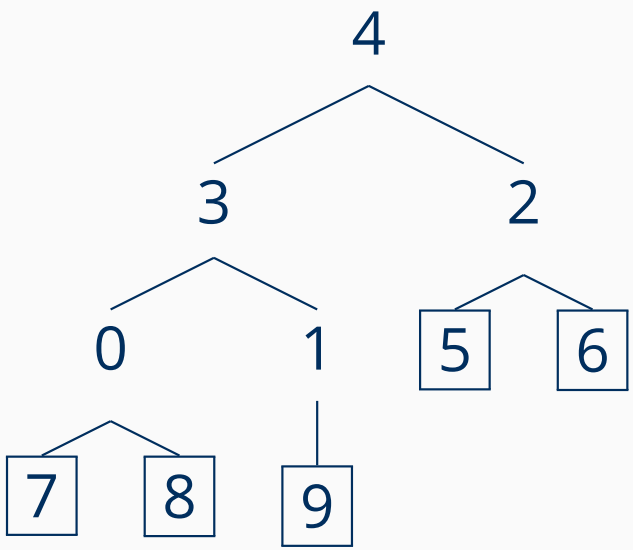
AUFGABE 2

Fortsetzung zur Vollständigkeit

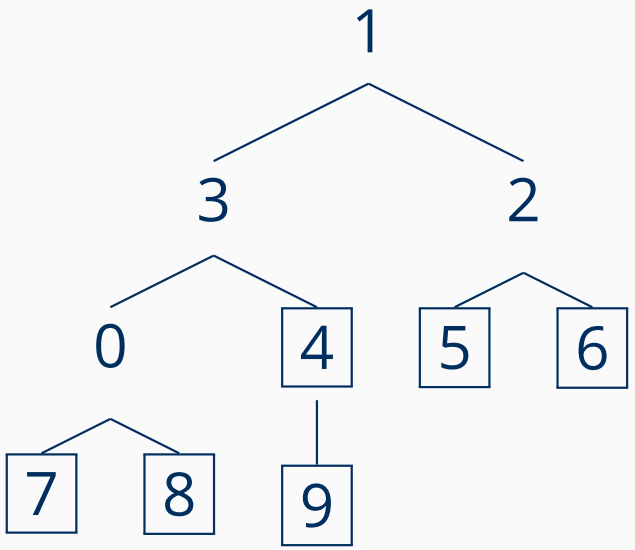
Tausch →



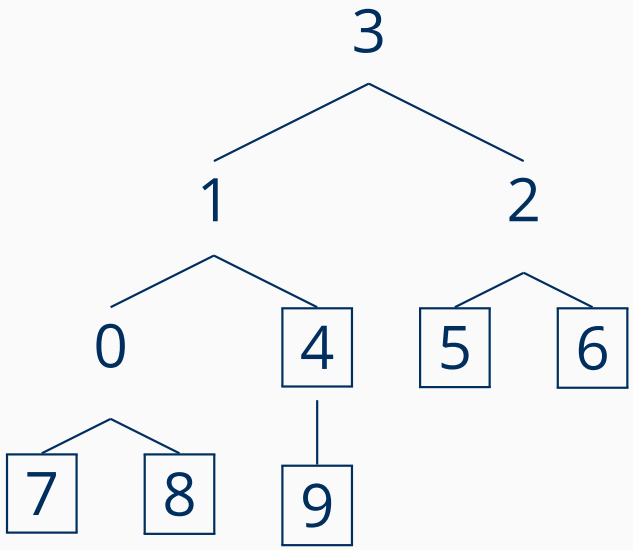
$s(2)$ →



Tausch →



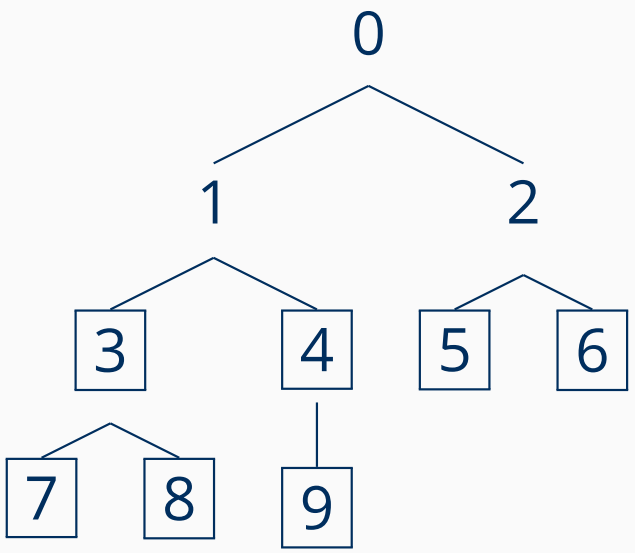
$s(1)$ →



AUFGABE 2

Fortsetzung zur Vollständigkeit

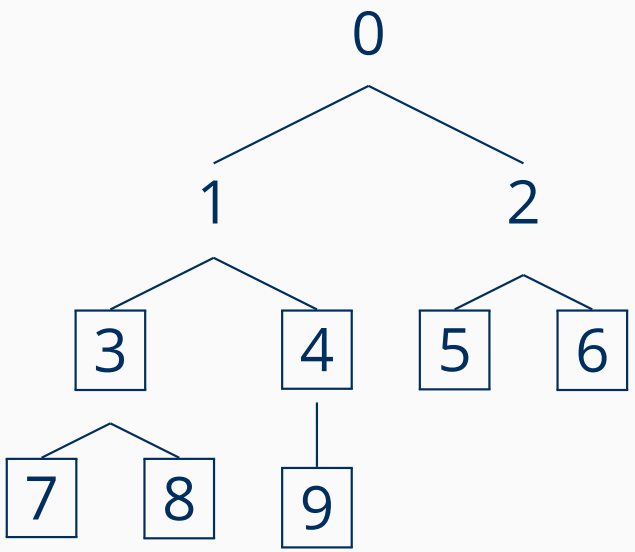
Tausch $\longrightarrow$



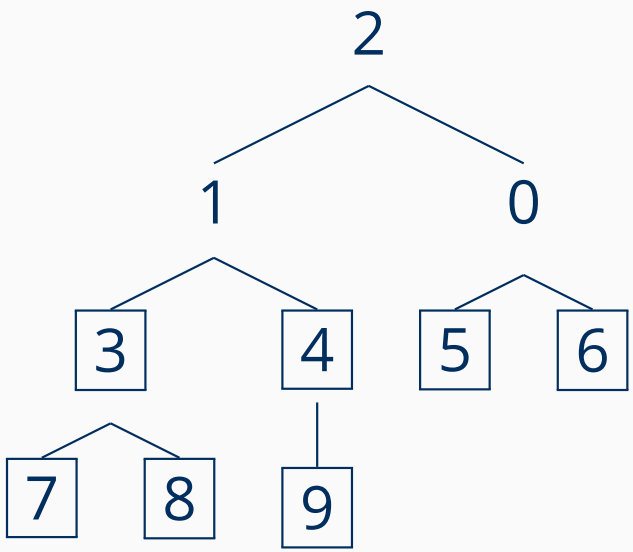
AUFGABE 2

Fortsetzung zur Vollständigkeit

Tausch $\longrightarrow$



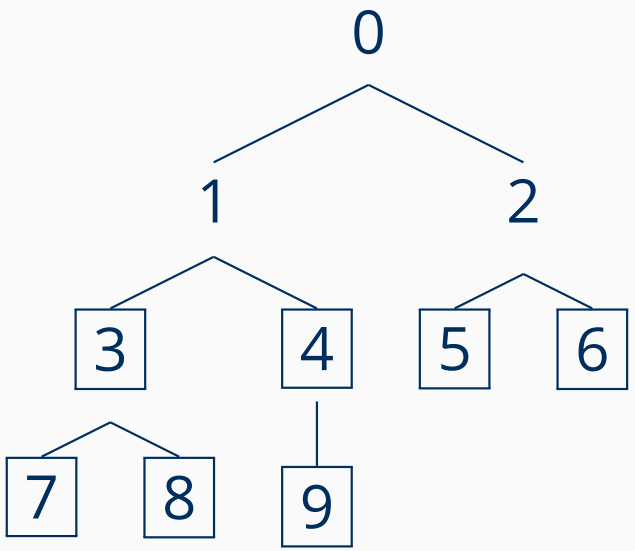
$s(0)$ $\longrightarrow$



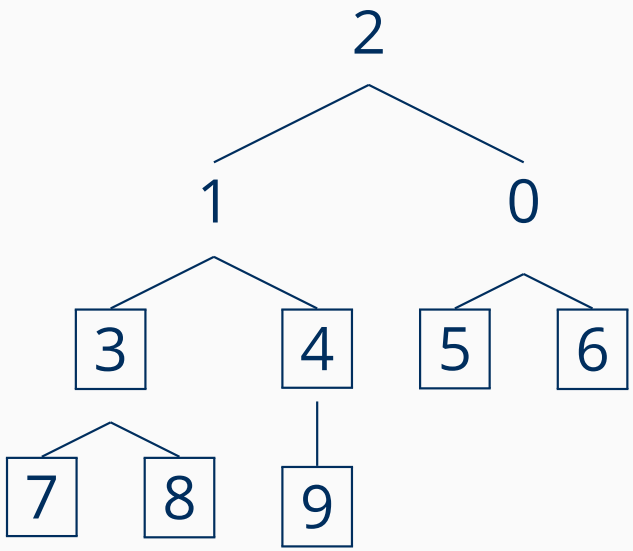
AUFGABE 2

Fortsetzung zur Vollständigkeit

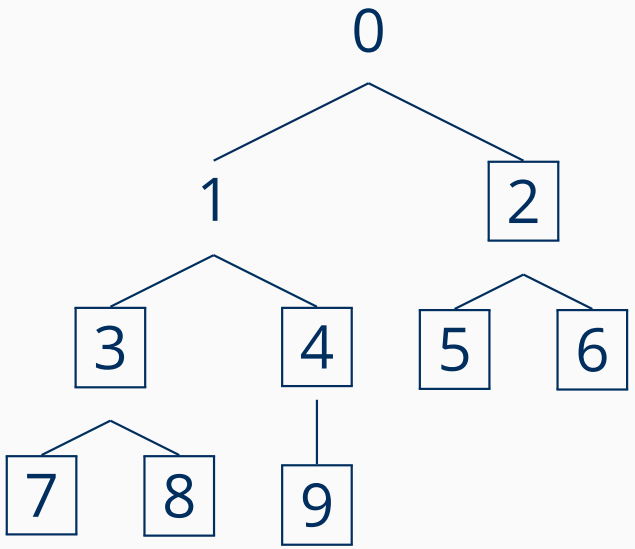
Tausch →



$s(0)$ →



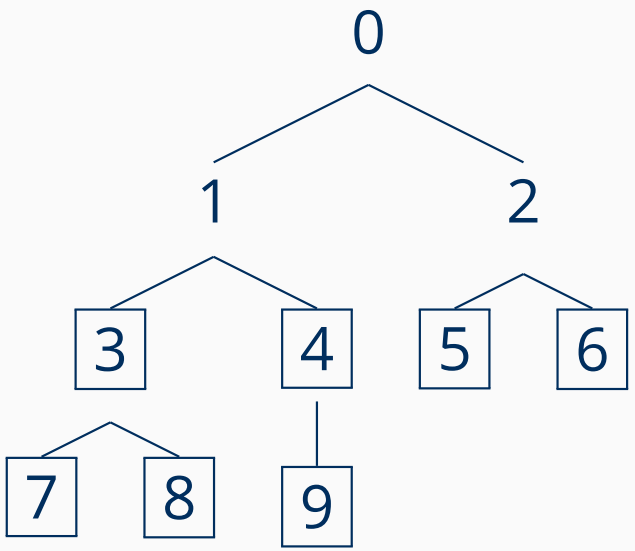
Tausch →



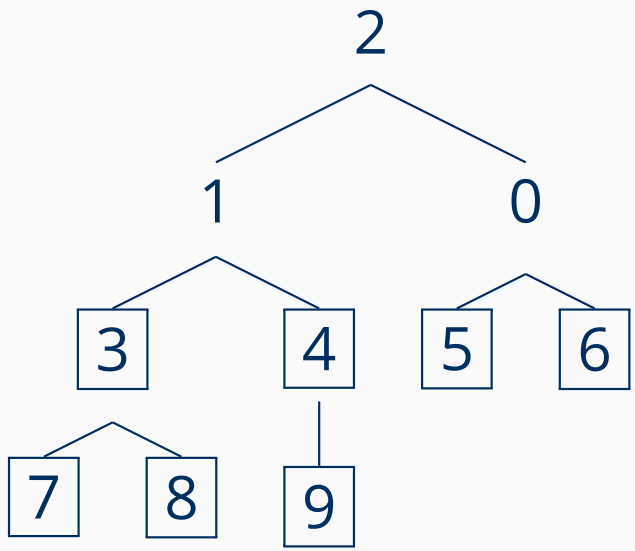
AUFGABE 2

Fortsetzung zur Vollständigkeit

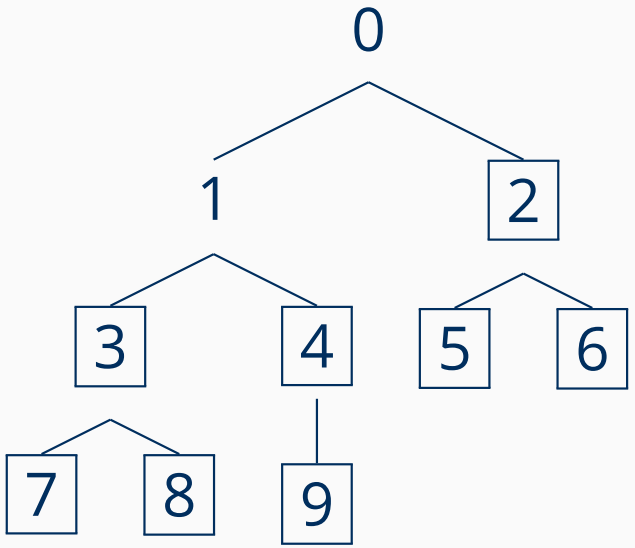
Tausch →



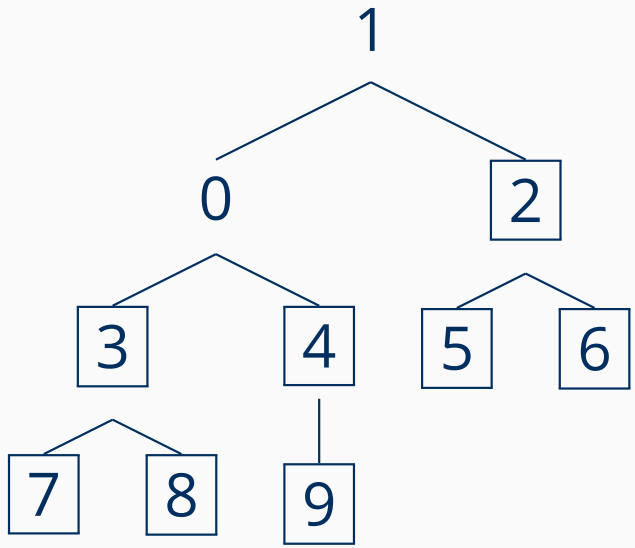
$s(0)$ →



Tausch →



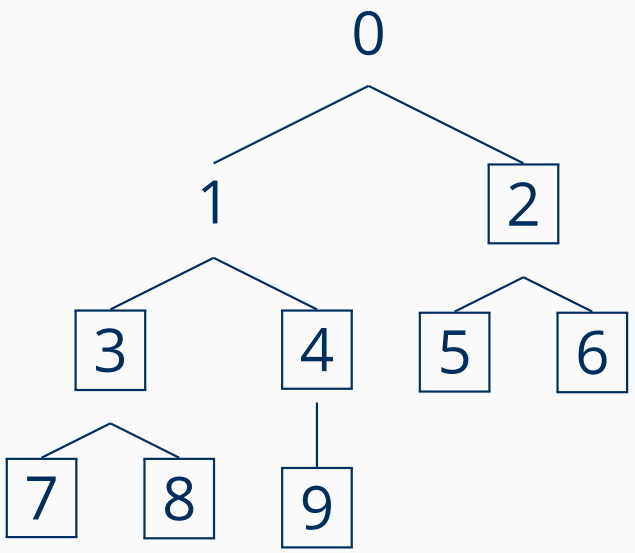
$s(0)$ →



AUFGABE 2

Fortsetzung zur Vollständigkeit

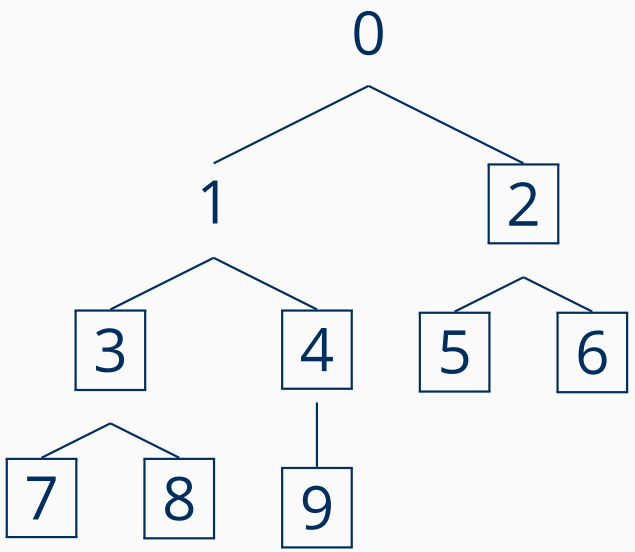
Tausch $\longrightarrow$



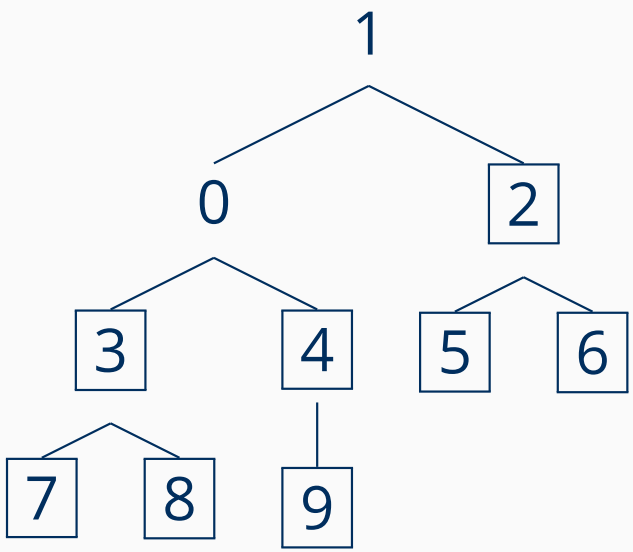
AUFGABE 2

Fortsetzung zur Vollständigkeit

Tausch $\longrightarrow$



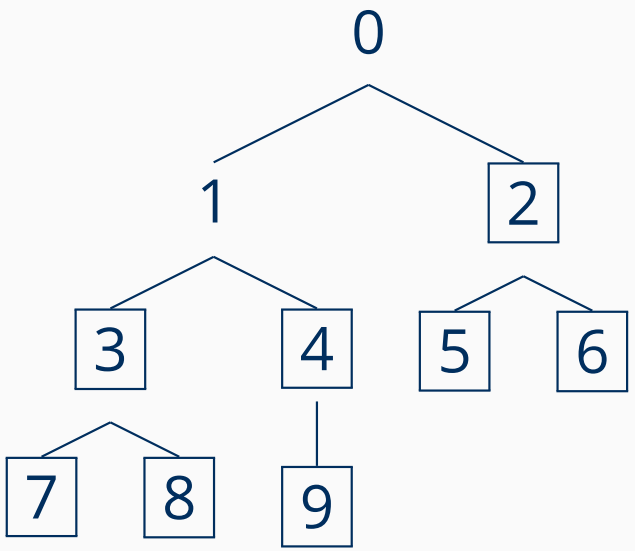
$s(0)$ $\longrightarrow$



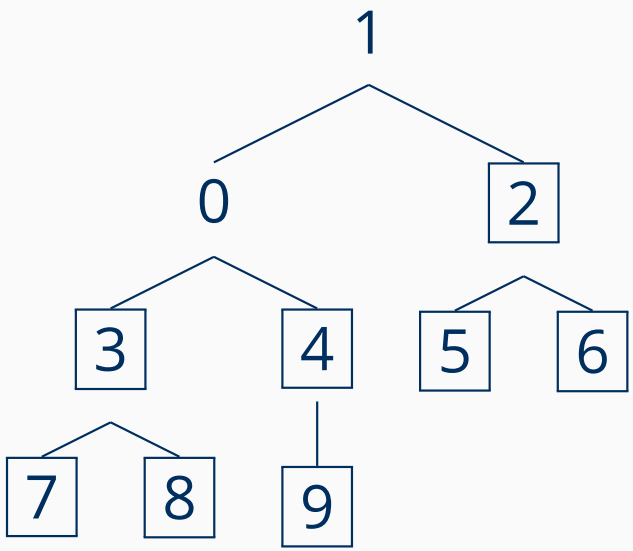
AUFGABE 2

Fortsetzung zur Vollständigkeit

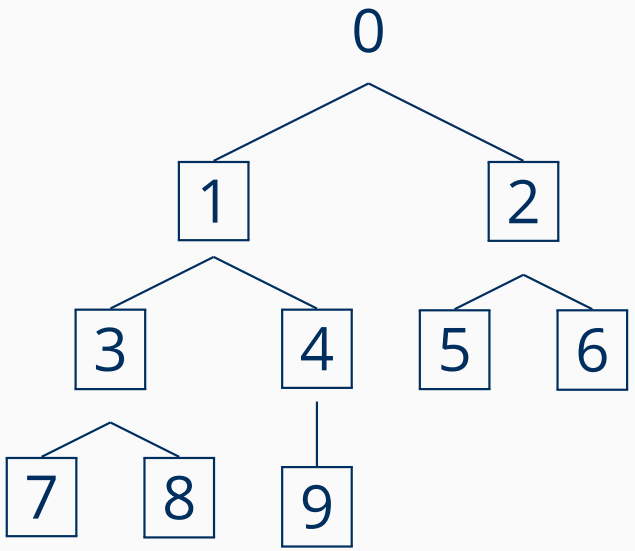
Tausch →



$s(0)$ →

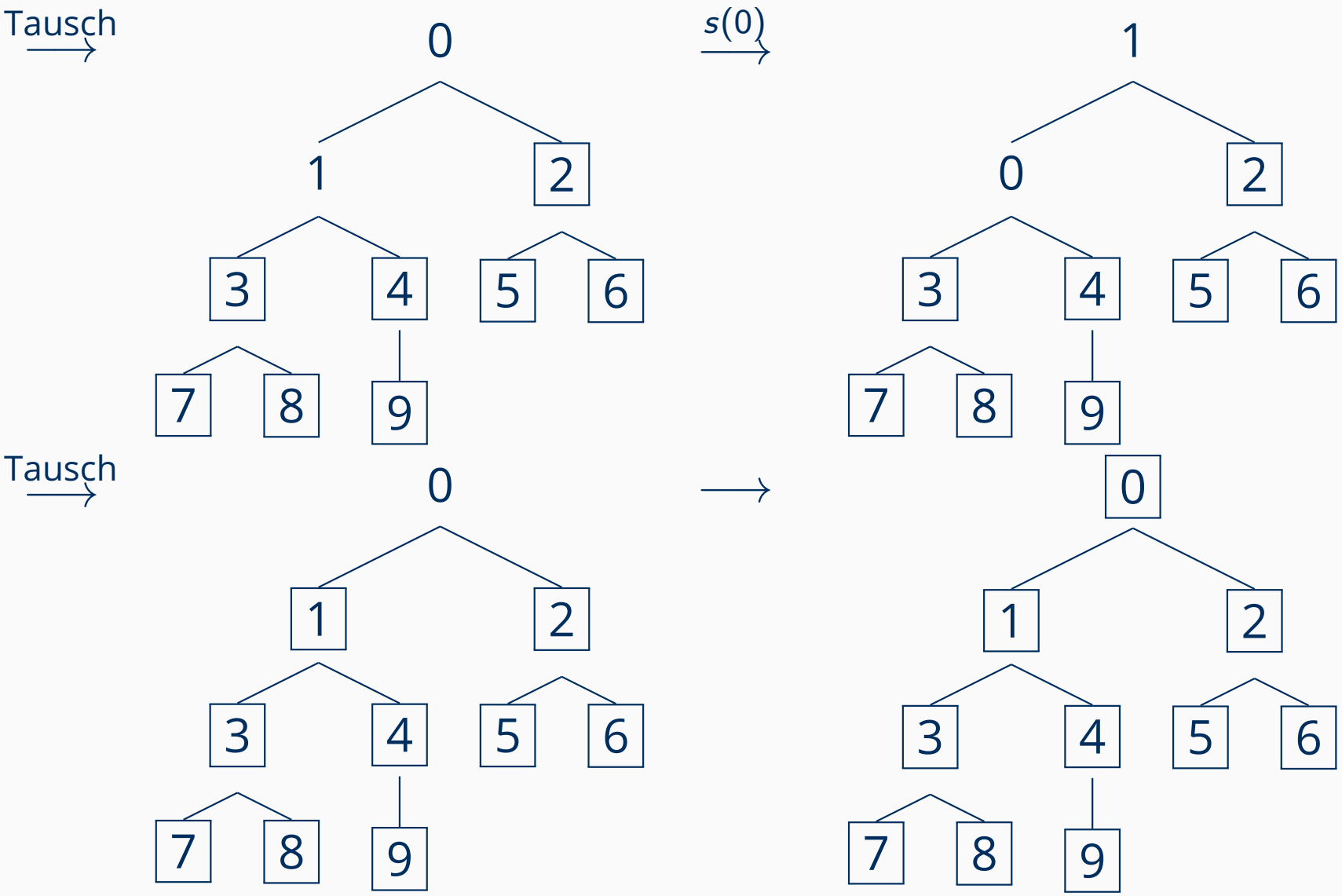


Tausch →



AUFGABE 2

Fortsetzung zur Vollständigkeit



AUFGABE 2

sortierte Liste:

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

FERTIG ☺