Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Friedrich Gretz, Benjamin Kaminski, Thomas Ströder

Tutoraufgabe 1 (Sortieralgorithmus):

Gegeben sei der folgende Sortieralgorithmus:

```
void sort(int E[]) {
  int i,j,m;
  for (i = 0; i < E.length; i++) {
    m = i;
    for (j = i + 1; j < E.length; j++) {
      if (E[j] <= E[m]) {
          m = j;
      }
    }
    int v = E[i];
    E[i] = E[m];
    E[m] = v;
  }
}
```

a) Nutzen Sie den gegebenen Algorithmus, um das folgende Array zu sortieren. Geben Sie den Zustand des Arrays nach jedem Durchlauf der äußeren Schleife an.

1	3	2	7	0	4	8	5	7	6

b) Geben Sie in wenigen Worten wieder, wie der gegebene Algorithmus funktioniert. In der Vorlesung wurden mehrere Sortierverfahren genannt (siehe Folien). Welcher Name passt zu diesem Algorithmus?

c) Ist der Sortieralgorithmus stabil? Falls dies nicht der Fall ist, geben Sie an, wie er angepasst werden muss, damit er stabil wird!

d) Welche Average-Case Laufzeit besitzt der gegebene Sortieralgorithmus für eine Eingabe der Länge n? Geben Sie die Komplexitätsklasse $\Theta(T_{sort}(n))$ für ein Array E mit Länge `n = E.length` an und begründen Sie Ihre Antwort.

Lösung:

a) In den folgenden Schritten sortiert der Algorithmus das Array:

1	3	2	7	0	4	8	5	7	6
0	3	2	7	1	4	8	5	7	6
0	1	2	7	3	4	8	5	7	6
0	1	2	7	3	4	8	5	7	6
0	1	2	3	7	4	8	5	7	6
0	1	2	3	4	7	8	5	7	6
0	1	2	3	4	5	8	7	7	6
0	1	2	3	4	5	6	7	7	8
0	1	2	3	4	5	6	7	7	8
0	1	2	3	4	5	6	7	7	8
0	1	2	3	4	5	6	7	7	8

b) Der Algorithmus sortiert das Array von vorne nach hinten indem er jeweils das kleinste Element aus dem noch zu sortierenden Teil sucht und dieses mit dem ersten Element dieses Bereiches tauscht. Dadurch wird von vorne nach hinten ein sortiertes Array aufgebaut.

Selectionsort ist ein geeigneter Name für dieses Verfahren, da jeweils das **kleinste** Element aus den übrigen Elementen **ausgesucht** wird.

c) Der Algorithmus ist nicht stabil. Um einen stabilen Algorithmus zu erhalten können wir das Vertauschen der Elemente durch ein Einfügen ersetzen, wie wir es von Insertionsort kennen. Zusätzlich muss der Vergleich der Elemente in Zeile 6 strikt sein.

d) Unabhängig von den jeweiligen Schlüsseln und ihrer Verteilung im Array wird die äußere Schleife n-fach durchlaufen, die innere Schleife jedesmal von der aktuellen Position bis zum Ende. Das Vertauschen der Werte geschieht in konstanter Zeit, da bei diesem Algorithmus das Aufschieben der Elemente entfällt. Es ergibt sich eine Laufzeitkomplexität von:

$$T_{sort(E)} \in \Theta(\sum_{i=0}^{n} i) = \Theta(n^2)$$

Tutoraufgabe 2 (Mergesort):

Sortieren Sie das folgende Array mithilfe von Mergesort aus der Vorlesung. Geben Sie dazu das Array nach jeder Merge-Operation an.

3	2	8	4	1	5	6	7	4

Lösung:

Die grau unterlegten Zeilen dienen nur zur Veranschaulichung, an welchen Stellen das Array aufgeteilt wird. Sie sind zur Lösung der Aufgabe nicht nötig.

[3 2 8 4 1 5 6 7 4]

[3 2 8 4 1] [5 6 7 4]

[3 2 8] [4 1] [5 6 7 4]

[3 2] [8] [4 1] [5 6 7 4]

[3] [2] [8] [4 1] [5 6 7 4]

[2 3] [8] [4 1] [5 6 7 4]

[2 3 8] [4 1] [5 6 7 4]

[2 3 8] [4] [1] [5 6 7 4]

[2 3 8] [1 4] [5 6 7 4]

[1 2 3 4 8] [5 6 7 4]

[1 2 3 4 8] [5 6] [7 4]

[1 2 3 4 8] [5] [6] [7 4]

[1 2 3 4 8] [5 6] [7 4]

[1 2 3 4 8] [5 6] [7] [4]

[1 2 3 4 8] [5 6] [4 7]

[1 2 3 4 8] [4 5 6 7]

[1 2 3 4 4 5 6 7 8]

Tutoraufgabe 3 (Max- und Min-Heaps):

a) Bestimmen Sie, ob folgende Arrays Heaps (Max-Heaps) sind. Falls nicht, geben Sie an wo die Heapeigenschaft verletzt ist.

1.)

56	47	56	10	20	50	51	1	2	3	18

2.)

56	56	47	10	20	50	51	1	2	3	18

3.)

56	47	56	10	51	20	50	1	2	3	18

4.)

56	47	56	10	5	20	50	1	2	3	18

b) In der Vorlesung wurden so genannte Max-Heaps vorgestellt. D.h jedes Element ist größer/gleich seiner Kinder. Ein *Min-Heap* ist ein Heap bei dem jedes Element kleiner/gleich seiner Kinderelemente ist.
Bestimmen sie, ob die folgenden Arrays Min-Heaps sind, und falls nicht, geben sie an, wo die Heapeigenschaft verletzt ist

1.)

1	5	1	8	10	4	15	11	12	14	11

2.)

0	1	5	8	10	4	15	11	12	14	11

3.)

1	5	1	11	8	4	15	11	12	14	10

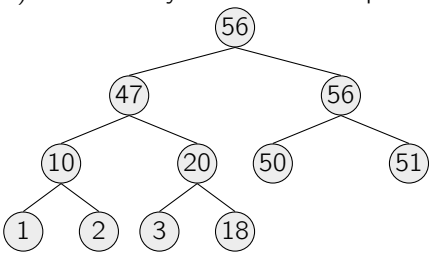
4.)

1	5	1	11	15	4	8	11	12	14	10

Lösung:

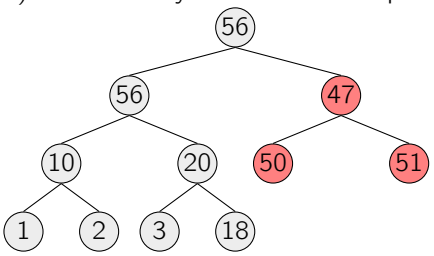
a) Die folgenden Heaps sind in den gegebenen Arrays repräsentiert:

1) Dieses Array ist ein Max-Heap.



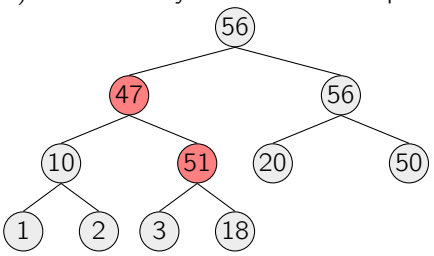
56 47 56 10 20 50 51 1 2 3 18

2) Dieses Array ist kein Max-Heap.



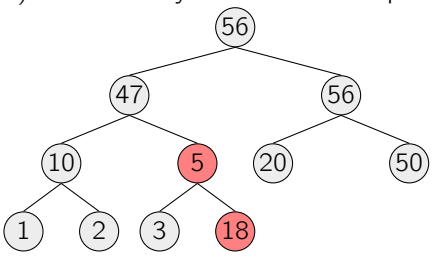
56 56 47 10 20 50 51 1 2 3 18

3) Dieses Array ist kein Max-Heap.



56 47 56 10 51 20 50 1 2 3 18

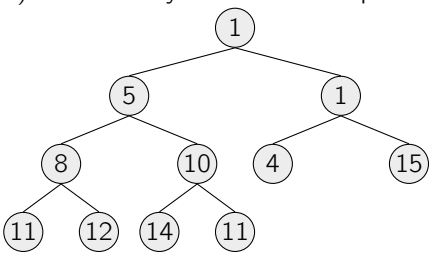
4) Dieses Array ist kein Max-Heap.



56 47 56 10 5 20 50 1 2 3 18

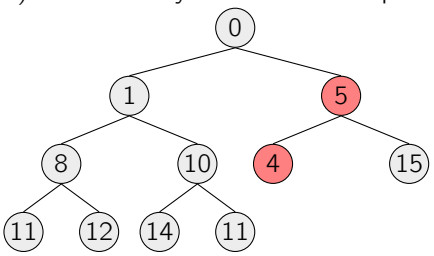
b) Die folgenden Heaps sind in den gegebenen Arrays repräsentiert:

1) Dieses Array ist ein Min-Heap.



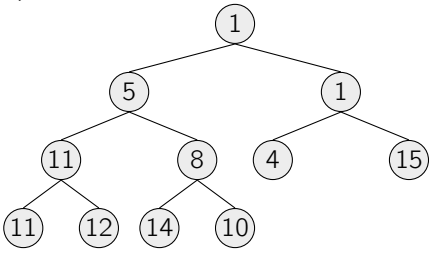
1 5 1 8 10 4 15 11 12 14 11

2) Dieses Array ist kein Min-Heap.



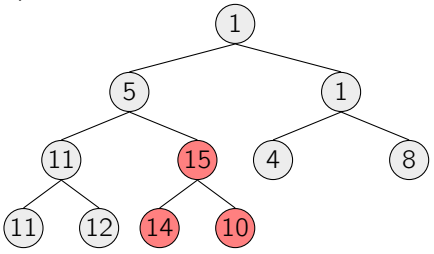
0 1 5 8 10 4 15 11 12 14 11

3) Dieses Array ist ein Min-Heap.



1 5 1 11 8 4 15 11 12 14 10

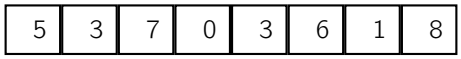
4) Dieses Array ist kein Min-Heap.



1 5 1 11 15 4 8 11 12 14 10

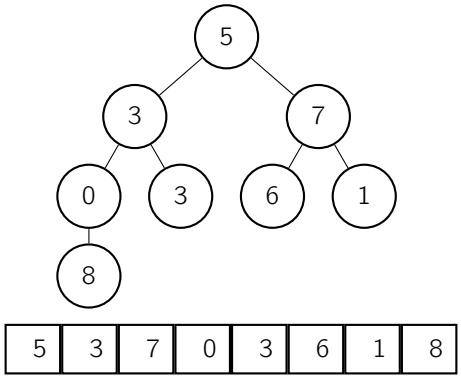
Tutoraufgabe 4 (Heapsort):

Sortieren Sie das folgende Array mithilfe von Heapsort aus der Vorlesung. Geben Sie dazu das Array nach jeder Swap-Operation an und geben Sie zum jeweils noch unsortierten Arraybereich zusätzlich die grafische Darstellung als Heap an.

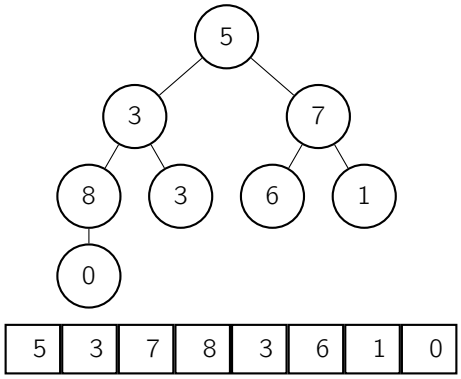


Lösung:

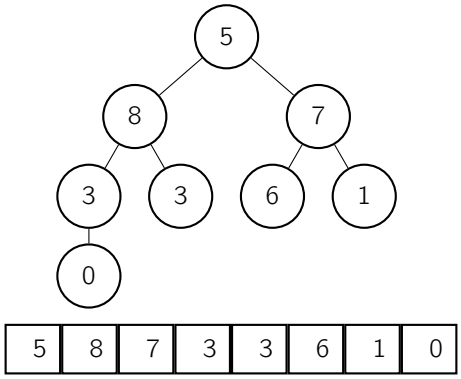
Schritt 0:



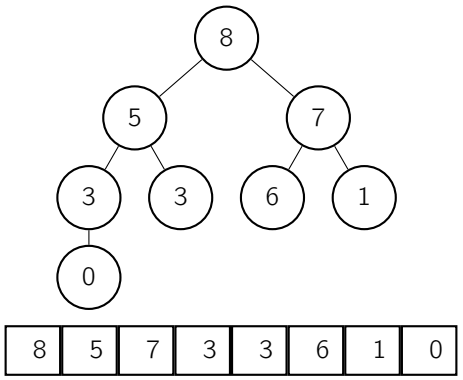
Schritt 1:



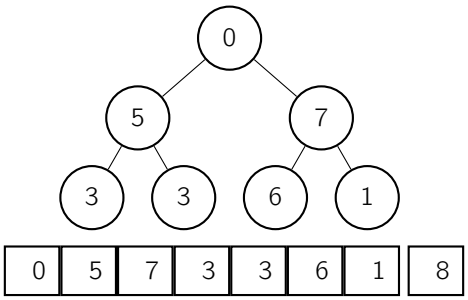
Schritt 2:



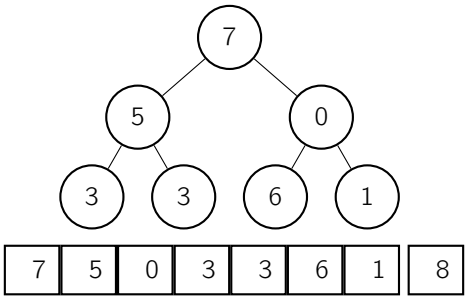
Schritt 3:



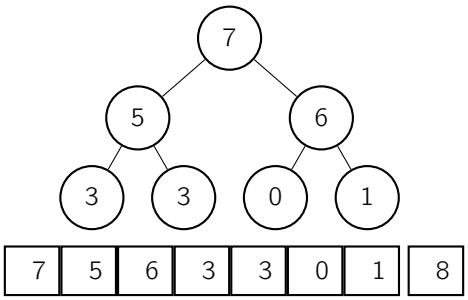
Schritt 4:



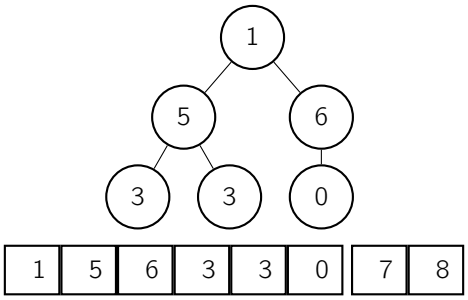
Schritt 5:



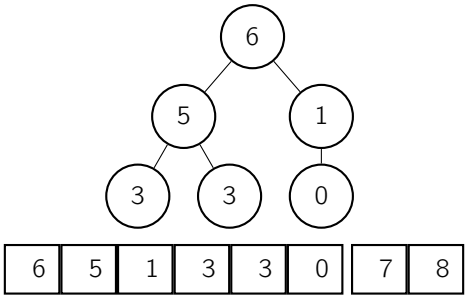
Schritt 6:



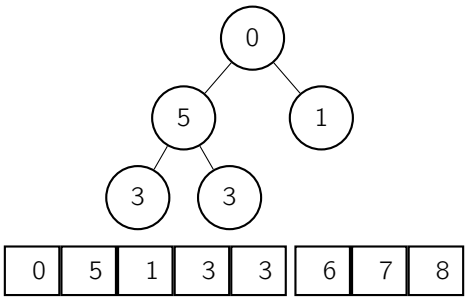
Schritt 7:



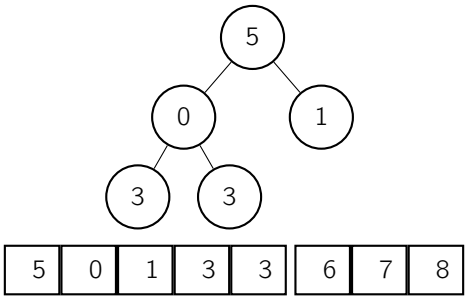
Schritt 8:



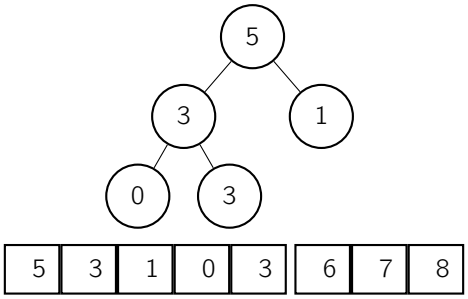
Schritt 9:



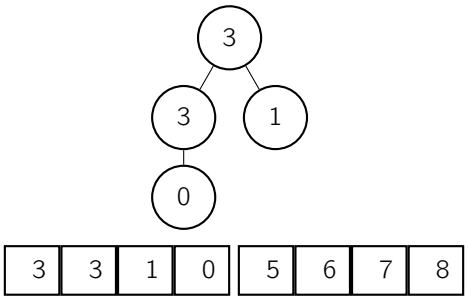
Schritt 10:



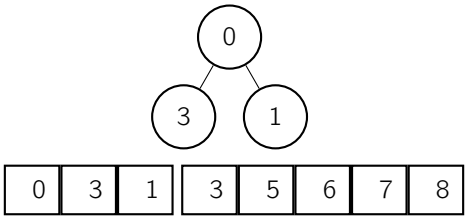
Schritt 11:



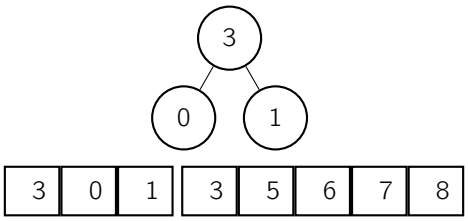
Schritt 12:



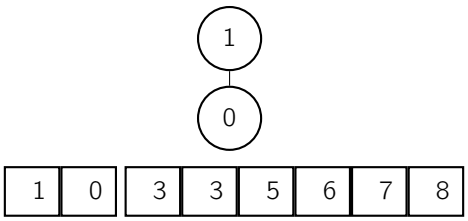
Schritt 13:



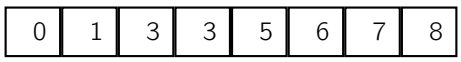
Schritt 14:



Schritt 15:



Schritt 16:



Tutoraufgabe 5 (Quicksort):

Sortieren Sie das folgende Array mithilfe von Quicksort aus der Vorlesung. Geben Sie dazu das Array nach jeder Partition-Operation an.

8	2	4	7	5	6	1	3

Lösung:

Die jeweils verwendeten Pivot-Elemente sind grau unterlegt.

8	2	4	7	5	6	1	3

1	2	**3**	7	5	6	8	4

1	**2**	3	7	5	6	8	4

1	2	3	**4**	5	6	8	7

1	2	3	4	5	6	**7**	8

1	2	3	4	5	**6**	7	8