

Algorithmen und Datenstrukturen - Hausübung 03

Gruppenmitglieder

- Emre Berber (2957148)
- Christoph Berst (2743394)
- Jan Braun (2768531)

Inhaltsverzeichnis

H1	1
a)	1
b)	1
c)	1
H2	1
a)	1
b)	1
c)	2
H3	2
a)	2
b)	2
c)	2

H1

a)

$$\Theta(n) = \frac{(n-1) \cdot (n+2)}{2} \quad \text{also} \quad O(n) = n^2$$

Je nachdem, ob in Zeile 4 vom Partition-Algorithmus (s. Foliensatz 02 - Seite 220-284) $<$ oder \leq steht, durchläuft i das Teilarray gar nicht oder komplett. Auf jeden Fall steht i nach Partition an einer der Enden des Teilarrays, weswegen der nächste Rekursionsaufruf unbalanciert mit (Anzahl der Elemente im aktuellen Teilarray) -1 Elementen geschieht. Das ganze geschieht $(n-1)$ mal, da man den letzten Fall bei dem das Teilarray nur noch aus einem Element besteht nicht betrachten muss. Wenn dieses Ablaufen der Elemente nun graphisch betrachten, haben wir nun ein Dreieck. Daher kommt auch das $\frac{(n+2)}{2}$. Die $(n+2)$ entstehen durch das aneinander legen der Dreiecke. Zur Erinnerung der letzte Rekursionsaufruf hat 2 Elemente.

b)

Teilarrayzerlegung s. Figure 1

Die fertige Liste sollte nach der Sortierung folgendermaßen aussehen:

[2, 3, 5, 7, 8, 10, 11, 16, 17, 19, 20]

c)

Best-Case:

[2, 11, 7, 19, 3, 16, 8, 20, 5, 17, 10]

23 Vergleiche

Worst-Case:

[2, 3, 5, 7, 8, 10, 11, 16, 17, 19, 20]

55 Vergleiche

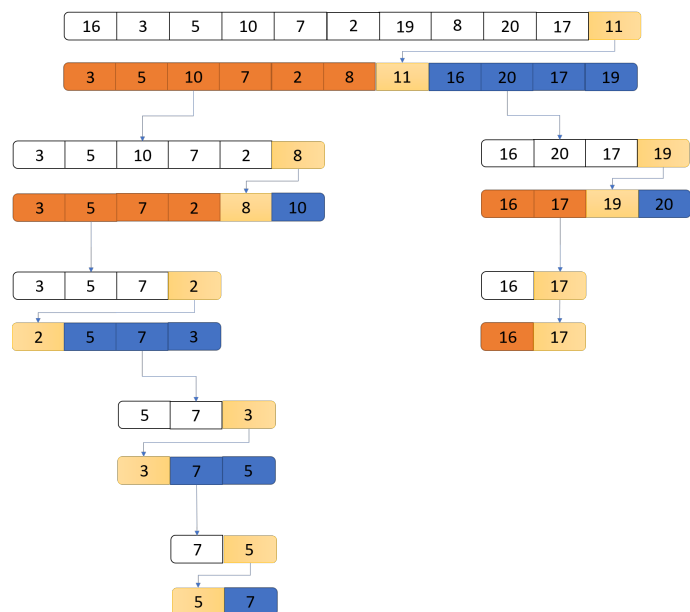


Figure 1: H1b - Quicksort - Teilarrayzerlegung

a)

```

Find-Modal-M(Array, left, right)
1  // left = 0
2  // right = Array.length - 1
3  WHILE true
4      m = left + (right - left) / 2
5      IF Array[m] > Array[m+1]
6          IF m = left
7              return m
8          right = m
9      ELSE
10         IF m = left
11             return m+1
12         left = m+1

```

Laufzeit bestimmt sich da dadurch, dass es sich durch die Zeilen 4, 8 und 12 das betrachtete Array immer halbiert wird.

$$f(n) \in O(\log n)$$

Sobald das betrachtete Array nur noch aus 2 Elementen besteht, wird das jeweils größere Element zurückgegeben.

Schleifeninvariante - **Korrektheitsbeweis!**

b)

```

BIGGEST-X()
1  RETURN Find-Modal-M(V, 0, V.length - 1)

```

c)

BIGGEST-Y()

1 RETURN Find-Modal-M(W, 0, BIGGEST-X())

Da hier einfach nur 2 mal Algorithmen mit $\log n$ ausgeführt werden, ist es immer noch in $O(\log n)$.

Die Punkten werden im Uhrzeigersinn beginnend mit dem Punkt mit dem kleinsten x-Wert aufgelistet. Die y-Werte in W könnten nun nach dem Hochpunkt m auch einen Tiefpunkt besitzen. Deshalb betrachten wir für den größten y-Wert nur den Halbkreis zwischen 0 und dem Punkt, mit dem größten x-Wert.

H3

a)

qwertz

b)

qwertz

c)

qwertz