Aufgabe 7

(a) Gegeben ist die Ausgabe der Methode **Partition** (s. Pseudocode), rekonstruieren Sie die Eingabe.

Konkret sollen Sie das Array A = (_, _, 1, _, _) so vervollständigen, dass der Aufruf Partition(A, 1, 5) die Zahl 3 zurückgibt und nach dem Aufruf gilt, dass A = (1, 2, 3, 4, 5) ist.

Geben Sie A nach jedem Durchgang der for-Schleife in **Partition** an.

```
 2  4  1  5  3  Eingabe
 2  4  1  5  3  zerlege
 2  4  1  5  3* markiere (i 4)
>2< 4  1  5  3  vertausche (i 0<>0)
 2 >4  1< 5  3  vertausche (i 1<>2)
 2  1 >4  5  3< vertausche (i 2<>4)
 2  1           zerlege
 2  1*          markiere (i 1)
>2  1<          vertausche (i 0<>1)
          5  4  zerlege
          5  4* markiere (i 4)
         >5  4< vertausche (i 3<>4)
 1  2  3  4  5  Ausgabe
```

(b) Beweisen Sie die Korrektheit von **Partition** (z. B. mittels einer Schleifeninvarianten)!

(c) Geben Sie für jede natürliche Zahl n eine Instanz I_n, der Länge n an, so dass QuickSort(I_n) $\Omega(n^2)$ Zeit benötigt. Begründen Sie Ihre Behauptung.

$I_n = 1, 2, 3, \ldots, n$

Die Methode **Partition** wird n mal aufgerufen, weil bei jedem Aufruf der Methode nur eine Zahl, nämlich die größte Zahl, abgespalten wird.

- Partition(A, 1, n)
- Partition(A, 1, $n - 1$)
- Partition(A, 1, $n - 2$)
- Partition(A, 1, $\ldots$)
- Partition(A, 1, 1)

In der For-Schleife der Methode Partition wird bei jeder Wiederholung ein Vertauschvorgang durchgeführt (Die Zahlen werden mit sich selbst getauscht.)

```
 1  2  3  4  5  6  7  zerlege
 1  2  3  4  5  6  7* markiere (i 6)
>1< 2  3  4  5  6  7  vertausche (i 0<>0)
 1 >2< 3  4  5  6  7  vertausche (i 1<>1)
```

```
 1  2 >3< 4  5  6  7  vertausche (i 2<>2)
 1  2  3 >4< 5  6  7  vertausche (i 3<>3)
 1  2  3  4 >5< 6  7  vertausche (i 4<>4)
 1  2  3  4  5 >6< 7  vertausche (i 5<>5)
 1  2  3  4  5  6 >7< vertausche (i 6<>6)
 1  2  3  4  5  6     zerlege
 1  2  3  4  5  6*    markiere (i 5)
>1< 2  3  4  5  6     vertausche (i 0<>0)
 1 >2< 3  4  5  6     vertausche (i 1<>1)
 1  2 >3< 4  5  6     vertausche (i 2<>2)
 1  2  3 >4< 5  6     vertausche (i 3<>3)
 1  2  3  4 >5< 6     vertausche (i 4<>4)
 1  2  3  4  5 >6<    vertausche (i 5<>5)
 1  2  3  4  5        zerlege
 1  2  3  4  5*       markiere (i 4)
>1< 2  3  4  5        vertausche (i 0<>0)
 1 >2< 3  4  5        vertausche (i 1<>1)
 1  2 >3< 4  5        vertausche (i 2<>2)
 1  2  3 >4< 5        vertausche (i 3<>3)
 1  2  3  4 >5<       vertausche (i 4<>4)
 1  2  3  4           zerlege
 1  2  3  4*          markiere (i 3)
>1< 2  3  4           vertausche (i 0<>0)
 1 >2< 3  4           vertausche (i 1<>1)
 1  2 >3< 4           vertausche (i 2<>2)
 1  2  3 >4<          vertausche (i 3<>3)
 1  2  3              zerlege
 1  2  3*             markiere (i 2)
>1< 2  3              vertausche (i 0<>0)
 1 >2< 3              vertausche (i 1<>1)
 1  2 >3<             vertausche (i 2<>2)
 1  2                 zerlege
 1  2*                markiere (i 1)
>1< 2                 vertausche (i 0<>0)
 1 >2<                vertausche (i 1<>1)
```

(d) Was müsste Partition (in Linearzeit) leisten, damit QuickSort Instanzen der Länge n in $\mathcal{O}(n \cdot \log n)$ Zeit sortiert? Zeigen Sie, dass Partition mit der von Ihnen geforderten Eigenschaft zur gewünschten Laufzeit von QuickSort führt.

> **Exkurs: Master-Theorem**
>
> $$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$
>
> $a =$ Anzahl der Unterprobleme in der Rekursion
>
> $\frac{1}{b} =$ Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird

$f(n) =$ Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen

Dann gilt:

1. Fall: $T(n) \in \Theta(n^{\log_b a})$

falls $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$ für $\varepsilon > 0$

2. Fall: $T(n) \in \Theta(n^{\log_b a} \cdot \log n)$

falls $f(n) \in \Theta(n^{\log_b a})$

3. Fall: $T(n) \in \Theta(f(n))$

falls $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ für $\varepsilon > 0$ und ebenfalls für ein c mit $0 < c < 1$ und alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

Die Methode **Partition** müsste die Instanzen der Länge n in zwei gleich große Teile spalten ($\frac{n-1}{2}$).

Allgemeine Rekursionsgleichung:

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

Anzahl der rekursiven Aufrufe (a):

2

Anteil Verkleinerung des Problems (b):

um $\frac{1}{2}$ also $b = 2$

Laufzeit der rekursiven Funktion ($f(n)$):

n

Ergibt folgende Rekursionsgleichung:

$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

für $\varepsilon = 4$:
$f(n) = n \notin \mathcal{O}\left(n^{\log_2 2 - \varepsilon}\right)$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$$f(n) = n \in \Theta\left(n^{\log_2 2}\right) = \Theta(n)$$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$:

$$f(n) = n \notin \Omega\left(n^{\log_2 2 + \varepsilon}\right)$$

$\Rightarrow T(n) \in \Theta\left(n^{\log_2 2} \cdot \log n\right) = \Theta(n \cdot \log n)$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

Funktion Quicksort(A, $l = 1$, $r = A$.length)

```
if l < r then
    m = Partition(A, l, r);
    Quicksort(A, l, m − 1);
    Quicksort(A, m + 1, r);
end
```

Funktion Partition(A, int l, int r)

```
pivot = A[r];
i = l;
for j = l to r − 1 do
    if A[j] ≤ pivot then
        Swap(A, i, j);
        i = i + l;
    end
end
```

Funktion Swap(A, int l, int r)

```
temp = A[i];
A[i] = A[y];
A[j] = temp;
```