Aufgabe 5 (Sortierverfahren)

In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen aufsteigend sortiert werden. Das Feld habe n Elemente A[0] bis A[n-1]. Der folgende Algorithmus (in der Notation des Informatik-Duden) sei gegeben:

```
procedure quicksort(links, rechts : integer)
var i, j, x : integer;
begin
  i := links;
  j := rechts;
  if j > i then begin
    x := A[links];
    repeat
      while A[i] < x do i := i+1;
      while A[j] > x do j := j-1;
      if i < j then begin
        tmp := A[i]; A[i] := A[j]; A[j] := tmp;
        i := i+1; j := j-1;:
      end
    until i > j;
    quicksort(links, j);
    quicksort(i, rechts);
  end
end
```

Umsetzung in Java:

```
  public static void quicksort(int[] A, int links, int rechts) {
    int i = links;
    int j = rechts;
    if (j > i) {
      int x = A[links];
      do {
        while (A[i] < x) {
          i = i + 1;
        }
        while (A[j] > x) {
          j = j - 1;
        }
        if (i <= j) {
          int tmp = A[i];
          A[i] = A[j];
          A[j] = tmp;
          i = i + 1;
          j = j - 1;
        }
        // Java verfügt über keine do-until Schleife.
        // Wir verwenden eine do-while-Schelfe mit einem umgedrehten Test
        // unit i > j -> while (i <= j)
      } while (i <= j);
      quicksort(A, links, j);
      quicksort(A, i, rechts);
    }
  }
```

Der initiale Aufruf der Prozedur lautet:

```
quicksort(0,n-1)
```

(a) Sortieren Sie das folgende Feld der Länge 7 mittels des Algorithmus. Notieren Sie jeweils alle Aufrufe der Prozedur quicksort mit den konkreten Parameterwerten. Geben Sie zudem für jeden Aufruf der Prozedur den Wert des in Zeile 7 gewählten Elements an.

27 13 21 3 6 17 44 42

```
quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
quicksort(0, 1)
3 6 17 32 27 44 42
x: 3
quicksort(0, -1)
3 6 17 32 27 44 42
quicksort(1, 1)
3 6 17 32 27 44 42
quicksort(2, 2)
3 6 17 32 27 44 42
quicksort(3, 6)
3 6 17 32 27 44 42
x: 32
quicksort(3, 3)
3 6 17 27 32 44 42
quicksort(4, 6)
3 6 17 27 32 44 42
x: 32
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(5, 6)
3 6 17 27 32 44 42
x: 44
quicksort(5, 5)
3 6 17 27 32 42 44
quicksort(6, 6)
3 6 17 27 32 42 44
3 6 17 27 32 42 44
```

(b) Angenommen, die Bedingung $j > i$ in Zeile 6 des Algorithmus wird ersetzt durch die Bedingung $j \geq i$. Ist der Algorithmus weiterhin korrekt? Begründen Sie Ihre Antwort.

geht, dauert aber länger

```
quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
quicksort(0, 1)
3 6 17 32 27 44 42
x: 3
quicksort(0, -1)
3 6 17 32 27 44 42
```

```
quicksort(1, 1)
3 6 17 32 27 44 42
x: 6
quicksort(1, 0)
3 6 17 32 27 44 42
quicksort(2, 1)
3 6 17 32 27 44 42
quicksort(2, 2)
3 6 17 32 27 44 42
x: 17
quicksort(2, 1)
3 6 17 32 27 44 42
quicksort(3, 2)
3 6 17 32 27 44 42
quicksort(3, 6)
3 6 17 32 27 44 42
x: 32
quicksort(3, 3)
3 6 17 27 32 44 42
x: 27
quicksort(3, 2)
3 6 17 27 32 44 42
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(4, 6)
3 6 17 27 32 44 42
x: 32
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(5, 6)
3 6 17 27 32 44 42
x: 44
quicksort(5, 5)
3 6 17 27 32 42 44
x: 42
quicksort(5, 4)
3 6 17 27 32 42 44
quicksort(6, 5)
3 6 17 27 32 42 44
quicksort(6, 6)
3 6 17 27 32 42 44
x: 44
quicksort(6, 5)
3 6 17 27 32 42 44
quicksort(7, 6)
3 6 17 27 32 42 44
3 6 17 27 32 42 44
```

(c) Angenommen, die Bedingung $i \leq j$ in Zeile 11 des Algorithmus wird ersetzt durch die Bedingung $i < j$. Ist der Algorithmus weiterhin korrekt? Begründen Sie Ihre Antwort.

bleibt hängen

```
quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
```

```
7  quicksort(0, 1)
8  3 6 17 32 27 44 42
9  x: 3
```

(d) Wie muss das Feld A gestaltet sein, damit der Algorithmus mit der geringsten Anzahl von Schritten terminiert? Betrachten Sie dazu vor allem Zeile 7. Begründen Sie Ihre Antwort und geben Sie ein Beispiel.

Im Worst Case (schlechtesten Fall) wird das Pivotelement stets so gewählt, dass es das größte oder das kleinste Element der Liste ist. Dies ist etwa der Fall, wenn als Pivotelement stets das Element am Ende der Liste gewählt wird und die zu sortierende Liste bereits sortiert vorliegt. Die zu untersuchende Liste wird dann in jedem Rekursionsschritt nur um eins kleiner und die Zeitkomplexität wird beschrieben durch $\mathcal{O}(n^2)$. Die Anzahl der Vergleiche ist in diesem Fall $\frac{n\cdot(n+1)}{2} - 1 = \frac{n^2}{2} + \frac{n}{2} - 1$.

Die Länge der jeweils längeren Teilliste beim rekursiven Aufrufe ist nämlich im Schnitt $\frac{2}{n}\sum_{i=\frac{n}{2}}^{n-1} i = \frac{3}{4}n - \frac{2}{4}$ und die Tiefe der Rekursion damit in $\mathcal{O}(\log(n))$. Im Average Case ist die Anzahl der Vergleiche etwa $2 \cdot \log(2) \cdot (n+1) \cdot \log_2(n) \approx 1,39 \cdot (n+1) \cdot \log_2(n)$.

(e) Die rekursiven Aufrufe in den Zeilen 16 und 17 des Algorithmus werden zur Laufzeit des Computers auf dem Stack verwaltet. Die Anzahl der Aufrufe von quicksort auf dem Stack abhängig von der Eingabegröße n sei mit s(n) bezeichnet. Geben Sie die Komplexitätsklasse von s(n) für den schlimmsten möglichen Fall an. Begründen Sie Ihre Antwort.