

Einzelprüfung „Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft)“

Einzelprüfungsnummer 46115 / 2019 / Herbst

Thema 1 / Aufgabe 4

((Sortiervverfahren))

Stichwörter: Selectionsort

In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen *aufsteigend* sortiert werden. Das Feld habe n Elemente $A[1]$ bis $A[n]$. Der folgende Algorithmus sei gegeben:

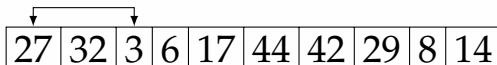
```
var A : array[1..n] of integer;

procedure selection_sort
var i, j, smallest, tmp : integer;
begin
  for j := 1 to n-1 do begin
    smallest := j;
    for i := j + 1 to n do begin
      if A[i] < A[smallest] then
        smallest := i;
    end
    tmp = A[j];
    A[j] = A[smallest];
    A[smallest] = tmp;
  end
end
```

- (a) Sortieren Sie das folgende Feld mittels des Algorithmus. Notieren Sie alle Werte, die die Variable *smallest* jeweils beim Durchlauf der inneren Schleife annimmt. Geben Sie die Belegung des Feldes nach jedem Durchlauf der äußeren Schleife in einer neuen Zeile an.

Lösungsvorschlag

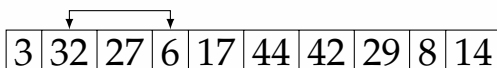
Ausgang



27	32	3	6	17	44	42	29	8	14
----	----	---	---	----	----	----	----	---	----

nach 1. Durchlauf ($j = 1$)

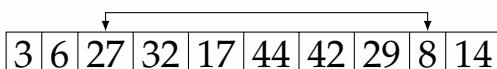
smallest: (1) 3



3	32	27	6	17	44	42	29	8	14
---	----	----	---	----	----	----	----	---	----

nach 2. Durchlauf ($j = 2$)

smallest: (2) 3 4



3	6	27	32	17	44	42	29	8	14
---	---	----	----	----	----	----	----	---	----

nach 3. Durchlauf ($j = 3$)

smallest: (3) 5 9

3	6	8	32	17	44	42	29	27	14
---	---	---	----	----	----	----	----	----	----

nach 4. Durchlauf ($j = 4$)

smallest: (4) 5 10

3	6	8	14	17	44	42	29	27	32
---	---	---	----	----	----	----	----	----	----

nach 5. Durchlauf ($j = 5$)

smallest: (5) -

3	6	8	14	17	44	42	29	27	32
---	---	---	----	----	----	----	----	----	----

nach 6. Durchlauf ($j = 6$)

smallest: (6) 7 8 9

3	6	8	14	17	27	42	29	44	32
---	---	---	----	----	----	----	----	----	----

nach 7. Durchlauf ($j = 7$)

smallest: (7) 8

3	6	8	14	17	27	29	42	44	32
---	---	---	----	----	----	----	----	----	----

nach 8. Durchlauf ($j = 8$)

smallest: (8) 10

3	6	8	14	17	27	29	32	44	42
---	---	---	----	----	----	----	----	----	----

nach 9. Durchlauf ($j = 9$)

smallest: (9) 10

3	6	8	14	17	27	29	32	44	42
---	---	---	----	----	----	----	----	----	----

fertig

3	6	8	14	17	27	29	32	42	44
---	---	---	----	----	----	----	----	----	----

- (b) Der Wert der Variablen *smallest* wird bei jedem Durchlauf der äußeren Schleife mindestens ein Mal neu gesetzt. Wie muss das Feld *A* beschaffen sein, damit der Variablen

smallest ansonsten niemals ein Wert zugewiesen wird? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Wenn das Feld bereits aufsteigend sortiert ist, dann nimmt die Variable *smallest* in der inneren Schleife niemals einen neuen Wert an.

- (c) Welche Auswirkung auf die Sortierung oder auf die Zuweisungen an die Variable *smallest* hat es, wenn der Vergleich in Zeile 9 des Algorithmus statt $A[i] < A[\text{smallest}]$ lautet $A[i] \leq A[\text{smallest}]$? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Der Algorithmus sortiert dann nicht mehr *stabil*, d. h. die Eingabereihenfolge von Elementen mit *gleichem* Wert wird beim Sortieren nicht mehr *bewahrt*.

- (d) Betrachten Sie den Algorithmus unter der Maßgabe, dass Zeile 9 wie folgt geändert wurde:

```
if A[i] > A[smallest] then
```

Welches Ergebnis berechnet der Algorithmus nun?

Lösungsvorschlag

Der Algorithmus sortiert jetzt absteigend.

- (e) Betrachten Sie die folgende *rekursive* Variante des Algorithmus. Der erste Parameter ist wieder das zu sortierende Feld, der Parameter n ist die Größe des Feldes und der Parameter $index$ ist eine ganze Zahl. Die Funktion $\text{min_index}(A, x, y)$ berechnet für $1 \leq x \leq y \leq n$ den Index des kleinsten Elements aus der Menge $\{A[x], A[x+1], \dots, A[y]\}$

```
procedure rek_selection_sort(A, n, index : integer)
var k, tmp : integer;
begin
if (Abbruchbedingung) then return;
  k = min_index(A, index, n);
  if k <> index then begin
    tmp := A[k];
    A[k] := A[index];
    A[index] := tmp;
  end
  (rekursiver Aufruf)
end
```

Der initiale Aufruf des Algorithmus lautet: $\text{rek_selection_sort}(A, n, 1)$

Vervollständigen Sie die fehlenden Angaben in der Beschreibung des Algorithmus für

- die Abbruchbedingung in Zeile 4 und

Lösungsvorschlag

`n = index` bzw `n == index`

Begründung: Wenn der aktuelle Index so groß ist wie die Anzahl der Elemente im Feld, dann muss / darf abgebrochen werden, denn dann ist das Feld sortiert.

- den rekursiven Aufruf in Zeile 11.

Lösungsvorschlag

`rek_selection_sort(A, n, index + 1)`

Am Ende der Methode wurde an die Index-Position *index* das kleinste Element gesetzt, jetzt muss an die nächste Index-Position (*index + 1*) der kleinste Wert, der noch nicht sortieren Zahlen, gesetzt werden.

Begründen Sie Ihre Antworten.

```
import static org.bsclangaul.helfer.Konsole.zeigeZahlenFeld;

public class SelectionSort {

    public static void selectionSort(int[] A) {
        int smallest, tmp;

        for (int j = 0; j < A.length - 1; j++) {
            System.out.println("\nj = " + (j + 1));
            smallest = j;
            for (int i = j + 1; i < A.length; i++) {
                if (A[i] < A[smallest]) {
                    smallest = i;
                }
            }
            tmp = A[j];
            A[j] = A[smallest];
            A[smallest] = tmp;
            zeigeZahlenFeld(A);
        }
    }

    public static void rekSelectionSort(int[] A, int n, int index) {
        int k, tmp;

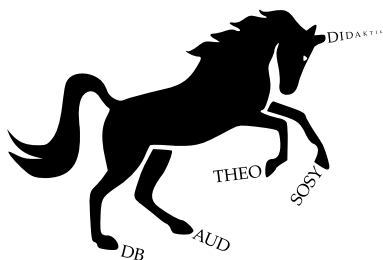
        if (index == n - 1) {
            return;
        }
        k = minIndex(A, index, n);
        if (k != index) {
            tmp = A[k];
            A[k] = A[index];
            A[index] = tmp;
        }
        rekSelectionSort(A, n, index + 1);
    }
}
```

```
public static int minIndex(int[] A, int x, int y) {
    int smallest = x;
    for (int i = x; i < y; i++) {
        if (A[i] < A[smallest]) {
            smallest = i;
        }
    }
    return smallest;
}

public static void main(String[] args) {
    int[] A = new int[] { 27, 32, 3, 6, 17, 44, 42, 29, 8, 14 };
    selectionSort(A);

    A = new int[] { 27, 32, 3, 6, 17, 44, 42, 29, 8, 14 };
    rekSelectionSort(A, A.length, 0);
    zeigeZahlenFeld(A);
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/SelectionSort.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/SelectionSort.java)



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net. Der TeX-Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: <https://github.com/bschlangaul-sammlung/examens-aufgaben/blob/main/Staatsexamen/46115/2019/09/Thema-1/Aufgabe-4.tex>