5. Felder

Zunächst wird Kapitel 6.1 des Buchs von Ratz, Scheffler, Seese durchgearbeitet.

Man kann Felder deklarieren wie in

```
int[][][] feld;
```

aber auch (nicht im Buch)

```
int feld[][][];
int[] feld[][];
   :
```

Das Partitionsproblem.

Eingabe: Ein Feld feld und ein x so daß x = feld[k] für ein k.

Ausgabe: Eine Umsortierung
von feld, so daß für ein $0 \le i \le n-1$

$feld[0], feld[1], \ldots feld[i] \le x$

und

$feld[i+1], \ldots, feld[n-1] \ge x$.

Grafisch:

Eingabe

$$feld = (f(0), f(1), \ldots, f(n-1))$$

$$x = f(k)$$

Ausgabe

$$feld' = (\underbrace{f'(0), f'(1), \ldots, f'(i)}_{\le x^0}, \overbrace{f'(i+1), \ldots f'(n-1)}^{\ge x^0})$$

Einige Beispiele:

Eingabe: $(1, 2, 3, 4, 5, 6)$, $x = 6$,

Ausgabe unverändert,

$i = 4$ oder $i = 5$.

Eingabe: $(1, 2, 3, 4, 4, 5)$, $x = 4$

Unverändert, $i = 3$ oder $i = 4$.

Eingabe $(1, 2, 3, 4, 4, 4)$, $x = 4$

Unverändert, $i = 3$, $i = 4$

oder auch $i = 5$.

Im Falle $i = 5$ ist die rechte „Hälfte" leer. Da $i \geq 0$ ist die „linke Hälfte" nie leer.

(5.4)

Das Problem ist leicht zu lösen:

1. Speichere die Elemente
   feld[i] $\leq x$ in einem
   neuen Feld. $feld_1$
2. Speichere die feld[i] $\geq x$, $\neq$
   in den Rest von $feld_1$.
3. Speichere $feld_1$ auf feld um.

Ungünstig, da doppelter Speicherplatz
und feld zunächst einmal
2-mal gelesen wird (feld.length
kann sehr groß sein > 1.000.000
zum Beispiel). Ziel: Umordnen nur
in dem array feld.

Idee des Algorithmus Partition am Beispiel:

feld = (5, 4, 3, 2, 7, 8, 1, 10, 11), x = 7

Zusätzliche Variablen i, j, deren Inhalt Adressen von feld darstellen.

$$(5, 4, 3, 2, 7, 8, 1, 10, 11)$$

$i = 0$ $\qquad$ $j = \text{feld.length} - 1$

Gehe von links mit i bis zum ersten Element von feld, das ≥ x ist.

$$(5, 4, 3, 2, 7, 8, 1, 10, 11)$$

$i = 4$ $\qquad$ j

Von rechts mit j bis zum ersten

Element $\leq x$.

$$(5, 4, 3, 2, 7, 9, 1, 10, 11)$$
$$\quad\quad\quad\quad\quad\quad\quad\uparrow_i \quad \uparrow_j$$

Vertausche feld[i] und feld[j]

$$(5, 4, 3, 2, 1, 9, 7, 10, 11)$$
$$\uparrow \; i = 6,\ j = 5$$

Dabei gehen wir mit i und j

gleich noch eins weiter.

Ein Tauschen reicht hier, aber

beachte etwa

$$(10, 9, 8, 7, 6, 5, 4, 3, 2, 1) \quad x = 5.$$

(5.7)

```
import Prog1Tools.IOTools;
public class Partition{
  public static void main(String[] args) {
  int [] feld;
  int i,k, j, x, n, temp;


  // Initialisierung des Feldes
  n = IOTools.readInteger("Bitte die Feldgroesse eingeben:");
  feld = new int[n];
  System.out.println();
  for (i=0; i<feld.length; i++)
    feld[i] = IOTools.readInteger("Bitte feld[" + i + "] eingeben: ");
  System.out.println();
  //
  //
  //
  k = IOTools.readInteger("Das k zur Partitionierung eingeben: ");
  if (k < 0 | k > feld.length - 1) {
     System.out.println("Das k = " + k + " liegt nicht im fraglichen Bereich.");
     return;
     }
  //
  // Ausgabe des Feldes
  //
  for (i=0; i<feld.length; i++)
    System.out.print(feld[i] + "  ");
  System.out.println();
  //
  // Ausgabe des k und des feld[k]
  //
  System.out.println(k+ " ist das k und feld["+k+"] ist " + feld[k]);
  //
  // Jetzt faengt das eigentliche Partition an.
  //
  i = 0;
  j = feld.length - 1;
  x = feld[k];
  //
  // Noch einmal zur Sicherheit die Ausgabe des Pivotelements.
  //
  System.out.println("Pivotelement: " + x);
  //
  //
```

5.8

```
   do {                          // Schleife (1)
     //
     //    Schleife (2)
     //
     while (feld[i] < x)
       i++;     // Suche von links her
     //
     //    Schleife (3)
     //
     while (feld[j] > x)
       j--;     // Suche von rechts her
     //
     //     Vertauschen (4)
     //
     if (i <= j) {
       temp = feld[i];
       feld[i] = feld[j];
       feld[j] = temp;
       i++;
       j--;
     //
     //
     }

  } while (i <= j);      // Ende der Schleife bei (5)
  //
  //
  //
  System.out.println("Indexueberschneidung bei i = " + i + " und j = " + j);
  //
  // Ausgabe
  //
  for (i=0; i<feld.length; i++)
    System.out.print(feld[i] + "  ");
  System.out.println();
  return;
  }
}
```

Einige Beispiele zum Algorithmus Partition.

feld = $(1, 2, 3, 4, 5, 6)$ $k = 2$, $x = 3$

$i = 0$ $j = 5$

Schleife (2)

$(1, 2, 3, 4, 5, 6)$

$i = 2$ j

Schleife (3)

$(1, 2, 3, 4, 5, 6)$

$i = 2$
$j = 2$

Vertauschen (4), da $i \leq j$.

$(1, 2, 3, 4, 5, 6)$

j i

$i - j = 2 \geq 0$ so Ende.

5.10

$feld = (1, 2, 4, 3, 5, 6)$, $k=2$, $x=4$

$i = 0$ $\quad$ $j = 5$

Schleifen (2), (3)

$$\begin{array}{cccccc} 1, & 2, & 4, & 3, & 5, & 6 \\ & & \uparrow & \uparrow & & \\ & & i & j & & \end{array}$$

Vertauschen (4)

$$\begin{array}{cccccc} 1, & 2, & 3, & 4, & 5, & 6 \\ & & \uparrow & \uparrow & & \\ & & j & i & & \end{array}$$

$i - j = 1 \geq 0$ Ende.

Das waren 2 Fälle, in denen die Schleife nach Ausführung von (4) verlassen wird. Verlassen ohne (4) ist auch möglich.

feld = (4, 2, 3, 5, 6)   k = 2, x = 3

i   j

(2), (3), (4)

(3, 2, 4, 5, 6)

i, j

j ≥ i ist wahr, also in Schleife (1), dann in (2).

(3, 2, 4, 5, 6)

j   i

Schleife (3) wird nicht gemacht, vertauscht (4) auch nicht also Ende da

$i - j \geq 1$.

Schleife (1) endet ohne … (5.12)

Vertauschen (4) auszuführen, wenn:

$$\left(\le x \;\ldots\; \le x \;\; < x \;\; < x \ldots < x \;\; > x \ldots > x \;\; \ge x \ldots \ge x \right)$$

i $\qquad$ j

Schleife (2)

$$\left(\le x \ldots \le x \quad < x \quad > x \right.$$

i

Schleife (3)

$$\left(\qquad < x \; > x \qquad \ldots \right)$$

j $\quad$ i

Ende, da $i - j \geq 1$, und

es findet kein Vertauschen (4) statt.

5.13

Der Versuch einer Verifikation.
Wir betrachten Schleife (1).

Für $l \geq 1$ sei

f_l = $feld_l$, i_l, j_l

wie bekannt. Für $l = 0$ seien
es die Anfangswerte. $x = x_0$ und
$k = k_0$ ist fest.

Zuerst überlegen wir einmal,
daß die Schleifen (2) und (3)
nicht über die Ränder von $feld_l$
hinauslaufen. Dazu folgende
Invariante:

Ist $i_l \leq j_l$, dann:

(Inv 1) Es gibt $i' \geq i_l$ und $f_l[i'] \geq x$.
Es gibt $j' \leq j_l$ und $f_l[j'] \leq x$.

(Inv1) gilt für i_0, j_0 mit

$i' = j' = k$.

Situation ist ja am Anfang

$$f_0(k)$$

$$(\quad \times \quad)$$

$$i_0 = 0 \leq k \leq j_0 = \text{feld.length} - 1$$

Jetzt zum Induktionsschluß. Gelte

(Inv1) nach dem l'ten,

also vor dem $l+1$'ten Lauf

von Schleife (1) mit

i_l, j_l, f_l, i' und j'.

Dabei ist $l \geq 0$ vorausgesetzt.

Wir betrachten den $\overbrace{l+1}^{\geq 1}$'ten Lauf. Also ist $i_l \leq j_l$.

Schleife (2) hält an mit einem $i_{l,1}$ wobei, wg. (Inv 1)

$$i_l \leq i_{l,2} \leq i'$$

(Gleichheit ist möglich). Damit ist $f_l(i_{l,1}) \geq x$.

Ebenso Schleife (3): wg. (Inv 1)

$$j_l \geq j_{l,3} \geq j'.$$

wobei $f_l(j_{l,3}) \leq x$.

Ist $i_{l,2} - j_{l,3} \geq 1$, so sind wir fertig. (Inv1) gilt mit

$$i' = i_{\ell,2} = i_{\ell+1}$$

$$j' = j_{\ell,3} = j_{\ell+1}$$

Ist aber $i_{\ell,2} \le j_{\ell,3}$, so wird $f_\ell(i_{\ell,2})$ und $f_\ell(j_{\ell,3})$ in (4) vertauscht.

Außerdem wird

$$i_{\ell+1} = i_{\ell,2} + 1$$

$$j_{\ell+1} = j_{\ell,3} - 1 .$$

Ist $i_{\ell+1} \le j_{\ell+1}$, so gilt (Inv1) mit

$$i' = j_{\ell,3}$$

$$j' = i_{\ell,2} .$$

Passiert, wenn $i_{\ell,2} = j_{\ell,3}$ oder aber $i_{\ell,2} = j_{\ell,3} - 1$

Ist aber $i_{\ell+1} > j_{\ell+1}$, so gilt (Inv1) von selbst.

5.17

Ein kleines Beispiel:

$$(3, 1, 2, 4, 1, 2) \qquad x = 3$$

$$i = 0$$
$$j = 2$$

i $\qquad\qquad$ j

(2), (3), (4)

$$(2, 1, 2, 4, 1, 3) \qquad i = 2$$
$$j = 3$$

(2), (3)

$$(2, 1, 2, 4, 1, 3)$$

i $\quad$ j

(4)

$$(2, 1, 2, 1, 4, 3)$$

j $\quad$ i

Die eigentliche Korrektheit von Schleife (1) folgt mit der Invariante

Es ist

(Inv 2) $\quad fe(0), \ldots, fe(\underbrace{i_e - 1}) \leq x$

und

$fe(\underbrace{i_e + 1}), \ldots$ Beachte (4), also nicht i_e, j_e!

$\ldots, fe(\text{feld.length} - 1) \geq x$

(Inv 2) gilt für f_0, i_0, j_0, da dann (formaljuristisch) die leeren Folgen von Elementen von f_0 angesprochen sind.

gelte (Inv 2) nach dem

l'ten Lauf also vor dem

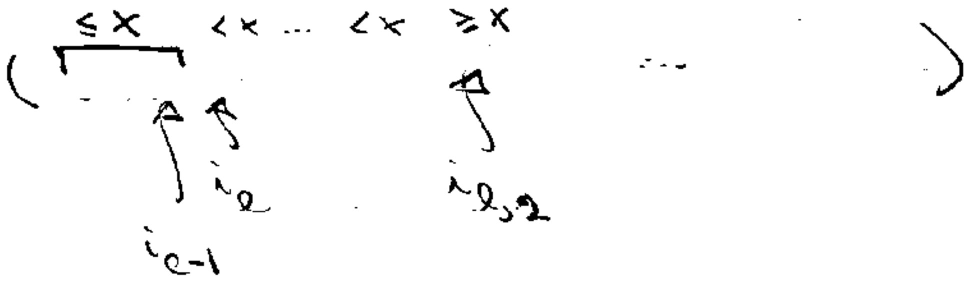
$l+1$'ten mit i_l, j_l, f_l.

Schleife (2) liefert als $i_{l,2}$ das

nächste $i' \geq i_l$ mit

$$f_l(i') \geq x.$$

Situation



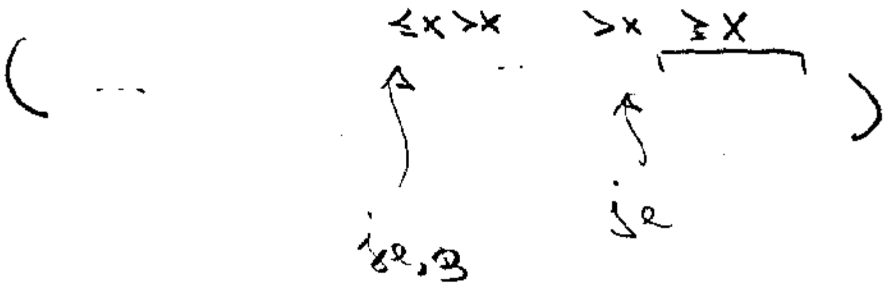
Dann ist jedenfalls

$$f_l(0), \dots, f_l(i_{l,1}-1) \leq x.$$

Schleife(3) liefert als $j_{e,3}$ das nächste $j' \le j_e$ mit

$$f_e(j') \le x.$$

Situation



Dann

$$f_e(\text{feld.length}-1), \ldots, f_e(j_{e,1}+1) \ge x.$$

Nun geht es weiter. Ist

$$i_{e,2} \le j_{e,3}$$

und (4) ausgeführt

Nach dem Vertauschen haben wir f_{e+1} vorliegen. Es ist

$$f_{e+1}(0), \ldots, f_{e+1}(i_{e,2}) \leq x^0$$

wg. Vertauschen (4).

$$f_{e+1}(\text{feld.length}-1), \ldots, f_{e+1}(j_{e,3}) \geq x^0$$

Außerdem wegen Hochzählen.

$$i_{e+1} = i_{e,2} + 1$$

$$j_{e+1} = j_{e,3} - 1$$

Also gilt die Invariante (Inv 2), da

eben

$$i_{e,2} = i_{e+1} - 1$$

$$j_{e,3} = j_{e+1} + 1.$$

Nun zur Quintessenz des ganzen.
Schleife (1) endet, mit dem
l'ten Lauf, wenn

$$i_l - j_l \geq 1$$

ist. Wegen (Inv 2) ist dann
im jeden Falle

$$f_l(0), \ldots, f_l(i_l - 1) \leq x$$

$$f_l(\text{feld.length}-1), \ldots, f_l(j_l+1) \geq x.$$

Da $i_l > j_l$ ist, ist

$$i_l - 1 \geq j_l.$$

Also steht das Gewünschte da:

5.23

$$f_e(0), \dots, f_e(j_e) \leq x$$

$$f_e(\text{feld.length}-1), \dots, f_e(j_e+1) \geq x.$$

Trennung an j_e.

Man könnte auch sagen

$$j_e + 1 \leq i_e$$

und analog wie oben mit

i_e und $i_e - 1$ statt j_e und j_e+1.

Schließlich zur # Durchläufe der Schleifen: (2) hält immer an wg. (Inv1). Ebenso (3). (1) hält an, da in (4) i++, j--. Wird nur (4) nicht gemacht, so ist i > j und (1) hält danach an.

Es werden i und j nur hoch- und runtergezählt.

Wie sieht es genau am Ende aus?

Dazu ganz genau eine weitere Invariante für Schleife (1):

Es ist

(Inv 3) $\quad i_e - j_e \leq 2.$

Beweis: Übungsaufgabe 8. Übung.

Ist $i_e - j_e = 2$, dann

ist $i_e + 1 = j_e + 1$ und wg. (Inv 2)

$$\ell_e(i_e - 1) = \ell_e(j_e + 1) = x.$$

Ist $i_e - j_e = 1$, dann $i_e - 1 = j_e$

oder $j_e + 1 = i_e$ und wir können

mit (Inv 2) nur sagen

$$\ell_e(i_e - 1) = \ell_e(j_e) \leq x$$

$$\ell_e(i_e) = \ell_e(j_e + 1) \geq x.$$

Eine alternative Möglichkeit

der Verifikation von Partition:

Induktion über die Größe des

Restes, angegeben durch $j_e - i_e$.

Dazu zeigen wir: Betreten von (1) mit einer Situation der Art

$$\text{feld} = (\underbrace{f(0), \dots,}_{\leq x}\ \overbrace{f(i), \dots, f(j)}^{j-i \geq 0}, \overbrace{\dots f(\text{length}-1)}^{\geq x})$$

mindestens eins $\geq x$

mindestens eins $\leq x$

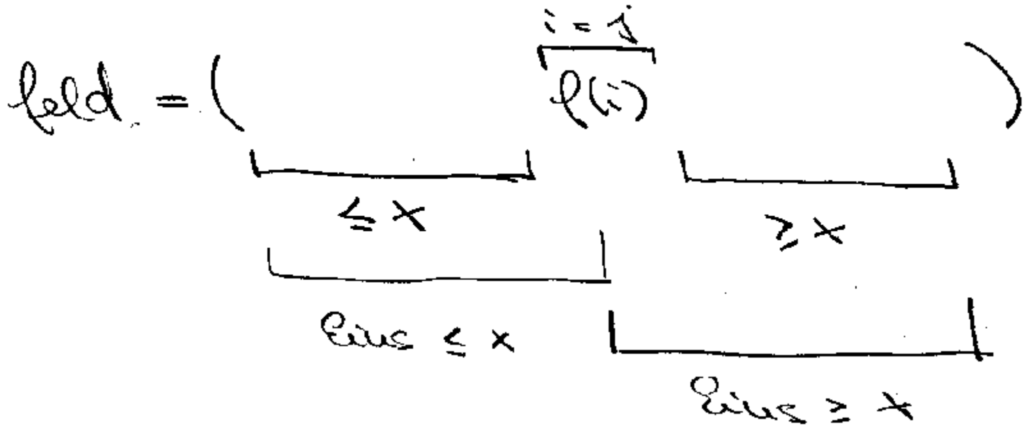
so hält (1) für ein $s \geq 1$ nach s Läufen an mit

$$\text{feld}_s = (\overbrace{f_s(0), \dots, f_s(j_s)}^{\leq x}, \overbrace{\dots f_s(\text{length}-1)}^{\geq x})$$

$j_s + 1$.

Ind. über $j - i \geq 0$. Ohne Id.-Anf.

Zeigen für alle (!)



gilt die Behauptung: Also,

betreten (1) ...

1. Fall $f(i) \not\leq x$

(2) hält mit $i+1$ (fehlerfrei, da mindestens eins $\geq x$ (!)).

(3) bewirkt keine Änderung an j

Da jetzt $i > j$ gilt Beh. mit $s = 1$. ...

2. Fall $f(i) >_* x$

Analog 1. Fall, i bleibt stehen, (3) endet mit $j-1$.

3. Fall $f(i) = x$.

(2), (3) ändern i und j nicht. Am Ende wird in (4) gezählt, Behauptung gilt.

Nun zum Induktionsschluß: $j - i \geq 0$

Nach Ind.-Vor gilt Beh. für alle (!) Situationen, in denen die Differenz um mindestens (!) 1 kleiner ist.

Also, betreten wieder (1). Dann (2).

1. Fall $f(i) < x$

Im ersten Lauf von (2) wird i um eins hochgezählt. „mindestens eines $\geq x$" gibt!

Betreten wir (1) gleich mit $i+1$ statt i, greift die Ind-Vor. Hier wird (2) mit derselben Situation betreten, die eben nach dem ersten Lauf von (2) vorliegt. Also folgt die Behauptung wegen der Ind.-Vor.

2. Fall $f(i) \geq x$, $f(j) > x$

ganz analog zum ersten Fall
j wird eins runtergezählt im
ersten Lauf von (3). Ind.-Vor
ist anwendbar.

3. Fall $f(i) \geq x$, $f(j) \leq x$

(2), (3) machen nichts. (4) tauscht
und zählt i eins hoch, j eins
runter. Sei

$$i_1 = i+1 , \quad j_1 = j-1 .$$

Ist $i_1 \ngeq j_1$, dann gilt die
Behauptung direkt. Denn dann $j = i+1$,
$j_1 = i_1 - 1$.

Ist aber $i_1 \leq j_1$ dann

$$j_1 - i_1 \lneq j - i.$$

und wir haben eine Situation vorliegen, daß die Ind.-Vor mit j_1, i_1 diese Feld noch dem Vertauschen anwendbar ist.

Man sieht: Diese Induktion über die Größe der noch zu bearbeitenden Struktur ist etwas kürzer als die Invarianten.

Anwendung von Partition beim Problem: k'-t. kleinstes Element.

Eingabe: Ein Feld feld von Zahlen und ein k mit $0 \le k \le$ f.length-1.

Ausgabe: Das Element von feld, das bei einer Sortierung an die k'te Stelle kommt (beziehungsweise bei k=0 mit dem Minimum).

feld = (1, 1, 3, 3, 2)

k = 2, Ausgabe 2

k = 1, Ausgabe 1

k = 3, Ausgabe 3

k = 4, Ausgabe 3.

(10, 20, 5)

k = 1, Ausgabe 10.

(5.33)

$k = 0$, minimum finden.

Einmal Feld durchgehen;
kleinstes Element mitführen.

$k = 1$,

kleinstes und nächstes Element
mitführen. 2 Elemente verwalten.

$k = 2$

3 kleinste Elemente verwalten.

$k = \lceil n/2 \rceil$ n = feld.length

$\lceil n/2 \rceil + 1$ Elemente verwalten.

Alternative: Sortieren.

Find für das Problem: k-tes Element.

```
import Prog1Tools.IOTools;
public class Find{
  public static void main(String[] args) {
  int [] feld;
  int i, j, k, x, n, links, rechts, temp;

  // Initialisierung des Feldes
  n = IOTools.readInteger("Bitte Feldgroesse eingeben: ");
  feld = new int[n];
  for (i=0; i<feld.length; i++)
    feld[i]=IOTools.readInteger("Bitte a[" + i + "] eingeben:");
  System.out.println();

  // Initialisierung von k
  System.out.println("Bitte k eingeben (0 <= k < " + n + ")");
  System.out.println("0 steht dabei fuer das kleinste Element");
  System.out.println((n-1) + " steht fuer das groesste Element");
  k = IOTools.readInteger("k = ");
  System.out.println();

  // Ausgabe
  for (i=0; i<feld.length; i++)
    System.out.print(feld[i] + "  ");
  System.out.println();

  // Find
  links = 0;
  rechts = feld.length-1;
```

```
while (links < rechts) {
  i = links;
  j = rechts;
  x = feld[k];
```

// Pivotelement für Partition

```
  // Aufteilen
  do {
    while (feld[i] < x)
      i++;     // Suche von links her
    while (feld[j] > x)
      j--;     // Suche von rechts her
    if (i <= j) {
      temp = feld[i];
      feld[i] = feld[j];
      feld[j] = temp;
      i++;
      j--;
    }
  } while (i <= j);
  if (j < k)
    links = i;
  if (k < i)
    rechts = j;
}
// Ausgabe
for (i=0; i<feld.length; i++)
  System.out.print(feld[i] + "  ");
System.out.println();
System.out.println("Das " + k + "-kleinste Element ist " + feld[k]);
return;
}
}
```

Wie Partition

Äußere Schleife

Ausgabe von feld[k].

Einige Beispiele.

5.36

feld = (1), $k = 0$

Kein Betreten der äußeren Schleife

Ausgabe von 1.

feld = (2,1), $k = 0$

$(2,1) \quad l = 0, \; r = 1$

$i = 0; \; j = 1$

$(1,2), \; i = 1, \; j = 0$

Es ist $k \leq i$, $k \geq j$

$l = 0, \; r = 0$

Ende und Ausgabe von 1.

$feld = (1\ 2\ 3) \quad k = 1$

$\downarrow$

$(1\ 2\ 3) \quad l = 0 \quad r = 2$

$\downarrow$ Partition auf $f(1) = 2$

$(1\ 2\ 3) \quad j = 0, \quad i = 2$

$j \quad\quad i$

Dann $l = 2, r = 0$ am Ende

und $f(1) = 2$

Zur Verifikation ist zu zeigen, daß wenn der l'te Lauf der letzte der äußeren Schleife ist

$$feld_e = (\overbrace{\quad\quad\quad\quad}^{\le f_e(k)}\ \underset{\underset{k\text{-te Stelle}}{\uparrow}}{f_e(k)}\ \overbrace{\quad\quad\quad\quad}^{\ge f_e(k)}).$$

Der zur Verifikation (Inv1) von Partition impliziert, daß wir nie über die Grenzen von feld laufen. Wir brauchen öfter (Inv2) von Partition

$$fe(0), \dots, fe(i_e-1) \le x$$

(Inv2) und Pivotelement

$$fe(j_e+1), \dots, fe(f.\mathrm{length}-1) \ge x .$$

Wir zeigen zunächst die Invariante (Find1) der äußeren Schleife:

links_ℓ $\quad$ rechts_ℓ

Ist $li_\ell \lneqq re_\ell$, so ist

(Find 1) $$li_\ell \leq k \leq re_\ell$$

Also k zwischen li_ℓ und re_ℓ.

(Find1) gilt am Anfang, bei $\ell = 0$.

gelte (Find1) bei $\ell-1$, wobei $\ell \geq 1$ ist.

Was passiert im ℓ-ten Lauf?

li_ℓ, re_ℓ werden in den if-statements

am Ende gesetzt. Wegen dem Ende

vom Partition ist dort $i \gneqq j$.

Linker Index $\quad$ Rechter Index

Sei nun $li_\ell \lneqq re_\ell$. Dann

ist nur ein if-statement ausgeführt

worden. Also ist

$k \geq i$ oder $k \leq j$. (nicht $k < i$; nicht $k > j$.)

Ist $k \geq i \gneq j$ dann ist mit Ind.-Vor.

$$links_\ell = i \leq k \leq rechts_{\ell+1} = rechts_\ell$$

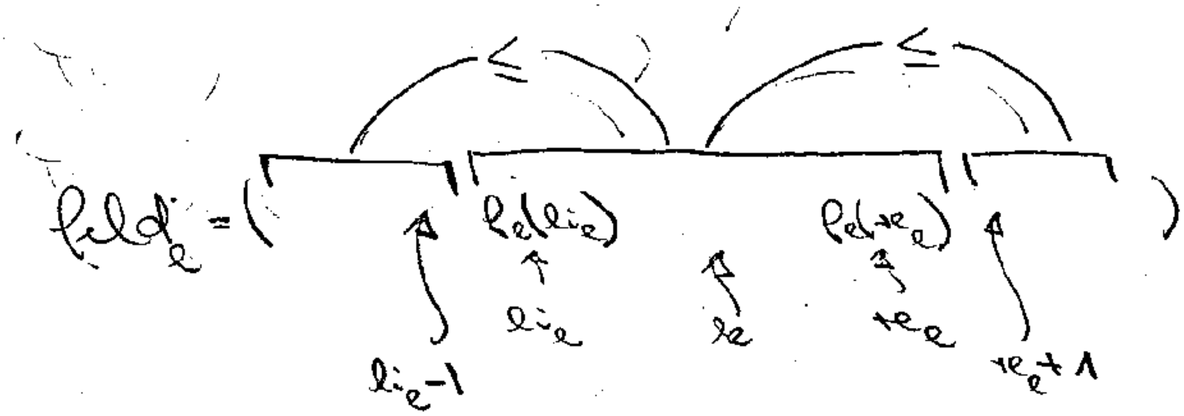
und (Find1) gilt. Ebenso für $k \leq j < i$. Dann ist

$$rechts_\ell = j \geq k \geq links_{\ell+1} = links_\ell.$$

und (Find1) gilt.

Zur Invariante (Find2) der äußeren Schleife:

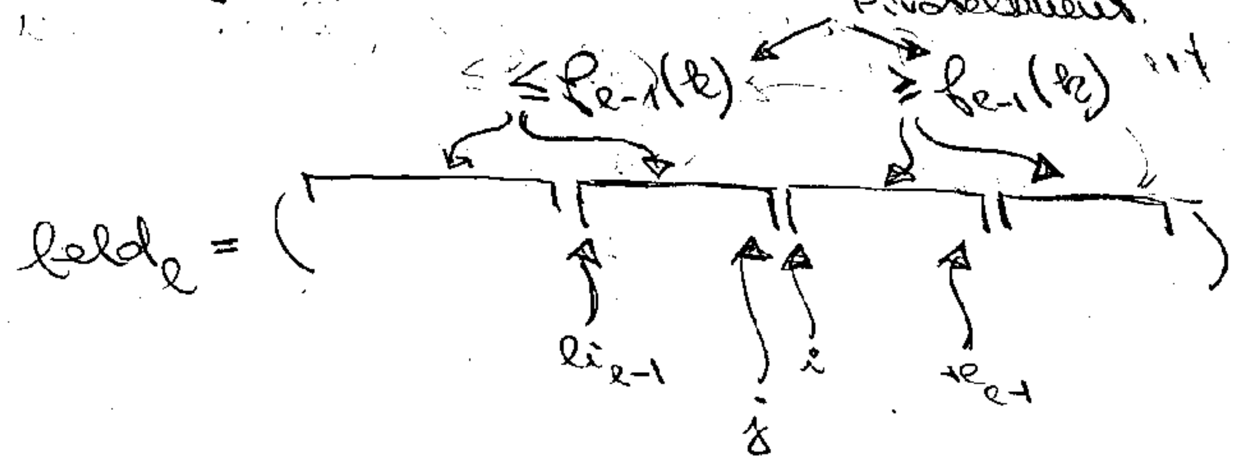
(Find2)



(Find 2) gilt mit $l=0$. Gilt (Find2) mit $l-1$ wobei $l \ge 1$. Wird die äußere Schleife betreten ist mit (Find1)

$$li_{l-1} \le k \le re_{l-1}$$

Am Ende von Partition liegt $feld_l$ vor. Es ist wegen Ind.-Vor. und (Inv 2) von Partition



Da $links_l = i$ werden kann oder $rechts_l = j$, gilt (Find 2).

Anzahl Durchläufe der äußeren Schleife endlich, da am Ende von Partition $i \geq j$ ist. Dann gilt

$$i \geq j \geq k \text{ und rechts} = j$$

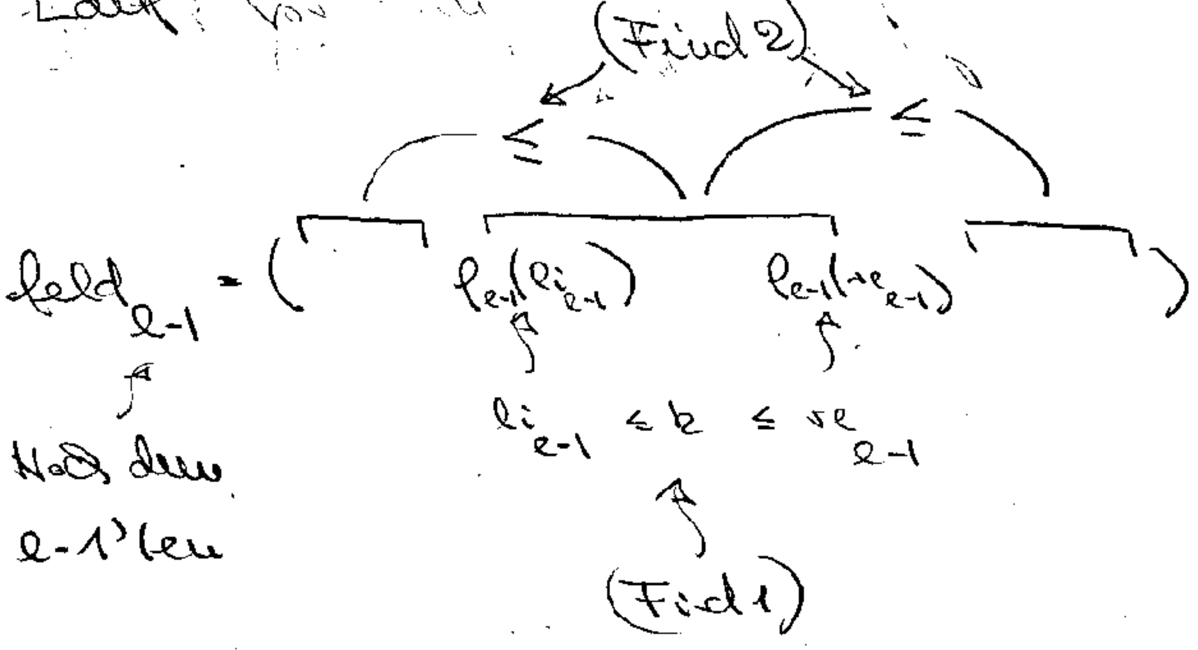
oder

$$j < k \leq i \text{ und links} = i$$

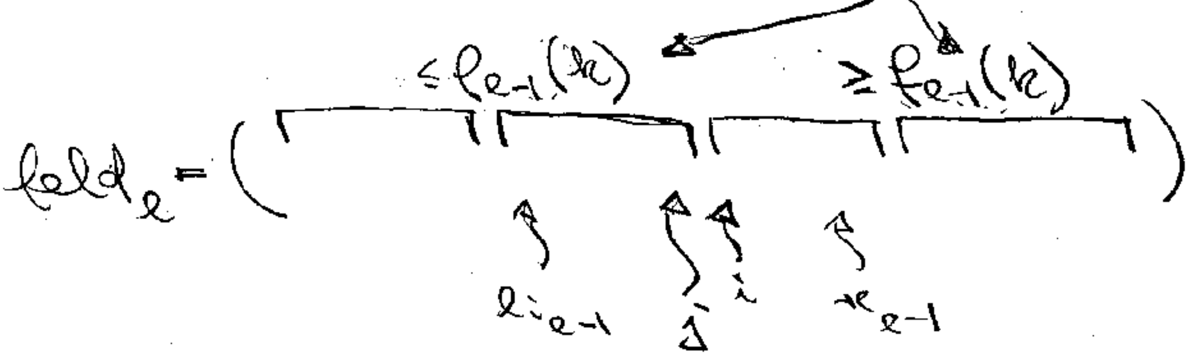
Damit wird rechts − links echt kleiner und die Anzahl der Läufe ist endlich.

Schließlich die Quintessenz des ganzen. Ist der l-te Lauf der letzte Lauf der äußeren Schleife.

Ist $l = 0$, dann ist die Verifikation erbracht. Ist $l > 0$, dann vor l'ten Lauf:



Am Ende von Partition



wegen (Inv 2) Partition und (Find 2). Da nun fertig sind wird

$$li_\ell \geq re_\ell .$$

1. Fall $li_\ell = re_\ell$

Da $i > j$ wird nur ein if-statement wirklich ausgeführt. Also

$$li_\ell = i \text{ und } re_\ell = re_{\ell-1}$$

oder

$$re_\ell = j \text{ und } li_\ell = li_{\ell-1} .$$

Im ersten Fall

$$j < k \text{ und } k \geq i$$

also wegen $re_\ell = li_\ell$ $k = li_\ell$.

(Find 2) ergibt die Verifikation.

Im zweiten Fall ist

$i > k$ und $k \leq j$

also $k = re_e$ und wieder mit (Find 2).

2. Fall: $li_e \geq re_e$

Dann sind beide If-statements ausgeführt worden. Dann ist

$j < k$ und $k < i$

Die Verifikation von Find folgt mit (Inv1) von Partition und (Find 2).

Die V.