

Algorithmen und Datenstrukturen

Aufgabe 1 Stack und Queue (Beispiellösung)

- | | | | |
|----------------------|--------------------------------------|----------------------------|--|
| a) | $\leftrightarrow (5, 8, 2, 3, 9)$ | b) | $\rightarrow (0, 9, 1, 3, 4) \rightarrow$ |
| (i) pop(S) = 5 | | (i) dequeue(Q) = 4 | |
| | $\leftrightarrow (8, 2, 3, 9)$ | | $\rightarrow (0, 9, 1, 3) \rightarrow$ |
| (ii) push(S , 8) | | (ii) enqueue(Q , 7) | |
| | $\leftrightarrow (8, 8, 2, 3, 9)$ | | $\rightarrow (7, 0, 9, 1, 3) \rightarrow$ |
| (iii) top(S) = 8 | | (iii) enqueue(Q , 5) | |
| | $\leftrightarrow (8, 8, 2, 3, 9)$ | | $\rightarrow (5, 7, 0, 9, 1, 3) \rightarrow$ |
| (iv) push(S , 1) | | (iv) dequeue(Q) = 3 | |
| | $\leftrightarrow (1, 8, 8, 2, 3, 9)$ | | $\rightarrow (5, 7, 0, 9, 1) \rightarrow$ |
| (v) pop(S) = 1 | | (v) isEmpty(Q) = false | |
| | $\leftrightarrow (8, 8, 2, 3, 9)$ | | $\rightarrow (5, 7, 0, 9, 1) \rightarrow$ |

Aufgabe 2 Insertion Sort (Beispiellösung)

- | | |
|--------------------|--------------------|
| a) (1, 9, 4, 3, 7) | b) (2, 7, 5, 1, 8) |
| (1) (9, 4, 3, 7) | (2) (7, 5, 1, 8) |
| (1, 9) (4, 3, 7) | (2, 7) (5, 1, 8) |
| (1, 4, 9) (3, 7) | (2, 5, 7) (1, 8) |
| (1, 3, 4, 9) (7) | (1, 2, 5, 7) (8) |
| (1, 3, 4, 7, 9) | (1, 2, 5, 7, 8) |

Aufgabe 3 Verifikation der Multiplikation durch Addition (Beispiellösung)

- a) Vor- und Nachbedingung von **Multiply**

$$\begin{aligned} \text{VOR} &= a \in \mathbb{Z} \wedge b \in \mathbb{Z} \\ \text{NACH} &= (c = a \cdot b) \end{aligned}$$

b) Schleifeninvariante für dritten Fall

$$P = (a \cdot b = c + (a - i + 1) \cdot b)$$

c) Abkürzung:

```
{VOR}
  if (a == 0 ∨ b == 0) {
    α
  }
  else if (a < 0) {
    β
  }
  else {
    γ
  }
{NACH}
```

- $\{VOR \wedge (a == 0 \vee b == 0)\} \alpha \{NACH\}$

Trivial, da $c = a \cdot b = 0$, falls $a = 0$ oder $b = 0$.

- $\{VOR \wedge \neg(a == 0 \vee b == 0) \wedge a < 0\} \beta \{NACH\}$

Vereinfachung der Vorbedingung:

$$VOR \wedge a < 0 \wedge b \neq 0$$

Allgemein bei $a < 0$:

$$a = -|a| = -1 \cdot |a| = -1 \cdot (-a),$$

$$c = a \cdot b = (-1 \cdot (-a)) \cdot b$$

Assoziativgesetz:

$$c = -1 \cdot ((-a) \cdot b).$$

Falls **Multiply** korrekt für positives a (Fall 3!), gilt auch hier $\{NACH\}$.

- $\{VOR \wedge \neg(a == 0 \vee b == 0) \wedge \neg(a < 0)\} \gamma \{NACH\}$

$\{C_0\}$

$c = 0;$

$i = 1;$

$\{C_1\}$

while ($i \leq a$) {

$\{C_2\}$

$c = c + b;$

$i = i + 1;$

$\{C_3\}$

}

$\{NACH\}$

Vereinfachung der Vorbedingung:

$$C_0 := (VOR \wedge a > 0 \wedge b \neq 0)$$

Bezeichne Schleifeninitialisierung mit γ_{init} und Schleifenkörper mit $\gamma_{\text{körper}}$.

- $C_1 \Rightarrow P$ (Korrektheit des Schleifeneintritts)

$$C_1 := (C_0 \wedge c = 0 \wedge i = 1)$$

Einsetzen von c und i in P :

$$c + (a - i + 1) \cdot b = 0 + (a - 1 + 1) \cdot b = a \cdot b$$

- $\{P \wedge (i \leq a)\} \gamma_{\text{körper}} \{P\}$ (Korrektheit des Schleifenkörpers)

$$C_2 := (P \wedge i \leq a)$$

Bezeichne *neue* Werte von c und i vorübergehend mit c' und i' :

$$c' = c + b$$

$$i' = i + 1$$

Invariante galt vorher:

$$a \cdot b = c + (a - i + 1) \cdot b$$

Nachher, unter Nutzung des alten Zustandes:

$$\begin{aligned} c' + (a - i' + 1) \cdot b &= (c + b) + (a - (i + 1) + 1) \cdot b \\ &= c + b + (a - i) \cdot b \\ &= c + (a - i + 1) \cdot b \\ &= a \cdot b \end{aligned}$$

- $P \wedge \neg(i \leq a) \Rightarrow NACH$ (Schleifenende, also verletzte Schleifenbedingung)

Es gilt $i = a + 1$ nach Konstruktion:

$$\begin{aligned} a \cdot b &= c + (a - i + 1) \cdot b \\ &= c + (a - (a + 1) + 1) \cdot b \\ &= c + (a - a - 1 + 1) \cdot b \\ &= c + 0 \cdot b \\ &= c \end{aligned}$$

Aufgabe 4 Verifikation von Selection Sort (Beispiellösung)

a) (i) SelectionSort

$$VOR_{\text{sort}} = n \geq 0$$

$$NACH_{\text{sort}} = A \text{ enthält Permutation der Originaldaten} \wedge A[0] \leq A[1] \leq \dots \leq A[n-1]$$

(ii) IndexOfMin

$$VOR_{\text{min}} = n > 0 \wedge 0 \leq j < n$$

$$NACH_{\text{min}} = j \leq i < n \wedge A[i] \leq A[k] \quad \forall k \in \{j, \dots, n-1\}$$

(iii) Swap ($\forall a, b \in \mathbb{N}$)

$$VOR_{\text{swap}} = n > 0 \wedge 0 \leq i, j < n \wedge A[i] = a \wedge A[j] = b$$

$$NACH_{\text{swap}} = A[i] = b \wedge A[j] = a \wedge A[k] \text{ unverändert} \quad \forall k \in \{0, \dots, n-1\} \setminus \{i, j\}$$

b) Konstruktion der Invariante:

- Permutation der Originaldaten in A laut $NACH_{sort}$.
- Sortierung der bereits bearbeiteten Indizes 0 bis $j-1$, also $A[0] \leq A[1] \leq \dots \leq A[j-1]$.
- Unsortierter Teil darf nichts Kleineres als $A[j-1]$ enthalten, also $A[j-1] \leq A[j..n-1]$.

Somit:

$$P = A \text{ enthält Permutation der Originaldaten} \wedge \\ A[0] \leq A[1] \leq \dots \leq A[j-1] \leq A[j..n-1]$$

Nach Ersatz der **for**-Schleife durch eine äquivalente **while**-Schleife:

```
{C0}
  j = 0;
{C1}
  while (j < n) {
{C2}
    i = IndexOfMin(A, j);
{C3}
    if (j ≠ i)
      Swap(A, i, j);
{C4}
    j = j + 1;
{C5}
  }
{C6}
```

{C₀} (Vorbedingung)

$$C_0 = VOR_{sort} = n \geq 0$$

{C₁} (Schleifeneintritt)

$$C_1 = C_0 \wedge j = 0 \wedge P \\ = n \geq 0 \wedge j = 0 \wedge P$$

Sei $n > 0$. (Sonderfall $n = 0$ unten!) Zu zeigen ist letztendlich:

$$C_6 = P \wedge \dots$$

{C₂} (ergibt Vorbedingung von **IndexOfMin**, VOR_{min})

$$C_2 = n > 0 \wedge 0 \leq j < n \wedge P$$

{C₃} (Nachbedingung von **IndexOfMin**, $NACH_{min}$; wird unten gezeigt)

$$C_3 = n > 0 \wedge 0 \leq j < n \wedge P \wedge \\ j \leq i < n \wedge A[i] \leq A[k] \quad \forall k \in \{j, \dots, n-1\}$$

{C₄} (Entweder bereits gegeben oder via Nachbedingung von **Swap**, $NACH_{swap}$)

$$C_4 = n > 0 \wedge 0 \leq j < n \wedge P \wedge \\ A[j] \leq A[(j+1)..(n-1)]$$

{C₅} („Verschiebung der Trennstelle“)

$$C_5 = n > 0 \wedge (0 \leq j < n \vee j \geq n) \wedge P$$

{C₆} (Nachbedingung im Fall $n > 0$)

$$\begin{aligned} C_6 &\stackrel{n>0}{\equiv} n > 0 \wedge j \geq n \wedge P \\ &= n > 0 \wedge j \geq n \wedge A \text{ enthält Permutation der Originaldaten} \wedge \\ &\quad A[0] \leq A[1] \leq \dots \leq A[j-1] \leq A[j..(n-1)] \\ &= n > 0 \wedge A \text{ enthält Permutation der Originaldaten} \wedge \\ &\quad A[0] \leq A[1] \leq \dots \leq A[n-1] \\ &= C_5 \wedge \neg(j < n) \\ &\Rightarrow NACH_{sort} \end{aligned}$$

{C₆} (Nachbedingung im Sonderfall $n = 0$)

$$\begin{aligned} C_6 &= n = 0 \wedge j = 0 \wedge P \\ C_6 &\stackrel{n=0}{\equiv} C_1 \wedge \neg(j < n) \\ &\Rightarrow NACH_{sort} \end{aligned}$$

c) Invariante:

$$P = A[i] \leq A[j..(k-1)]$$

Nach Ersatz der **for**-Schleife sowie Entfernen der Hilfsvariable A_i :

{C₀}

$$\begin{aligned} i &= j; \\ k &= j + 1; \end{aligned}$$

{C₁}

while ($k < n$) {

{C₂}

$$\begin{aligned} \text{if } (A[k] < A[i]) \\ i &= k; \end{aligned}$$

{C₃}

$$k = k + 1;$$

{C₄}

}

{C₅}

{C₀} (Vorbedingung)

$$C_0 = VOR_{min}$$

{C₁} (Schleifeneintritt)

$$C_1 = C_0 \wedge i = j \wedge k = j + 1 \wedge P$$

Sei $n > j + 1$. (Sonderfall $n = j + 1$ unten!)

{C₂} (Schleifeninvariante und Schleifenbedingung)

$$C_2 = n > 0 \wedge 0 < k < n \wedge \underbrace{A[i] \leq A[j..(k-1)]}_{=P}$$

{C₃} (Entweder bereits gegeben oder via Zuweisung $i = k$)

$$\begin{aligned} C_3 &= \dots \wedge A[i] \leq A[j..k] \\ &= \dots \wedge P \wedge A[i] \leq A[k] \end{aligned}$$

{C₄} („Verschiebung der Trennstelle“)

$$C_4 = n > 0 \wedge (0 < k < n \vee k \geq n) \wedge P$$

{C₅} (Nachbedingung im Fall $n > j + 1$)

$$C_5 \stackrel{j < n-1}{=} C_4 \wedge \neg(k < n) \Rightarrow NACH_{min}$$

{C₅} (Nachbedingung im Sonderfall $n = j + 1$)

$$C_5 \stackrel{j = n-1}{=} C_1 \wedge \neg(k < n) \Rightarrow NACH_{min}$$