# Musterlösungen
# zur Hauptklausur
# „1663 Datenstrukturen“

**10. August 2002**

## Aufgabe 1

(a)

Die Werte
$T(0) = 0 = 0 \cdot 0$,
$T(1) = 1 = 1 \cdot 1$,
$T(2) = 2T(1) - T(0) + 2 = 4 = 2 \cdot 2$,
$T(3) = 2T(2) - T(1) + 2 = 8 - 1 + 2 = 9 = 3 \cdot 3$
verleiten zu der Hypothese $T(n) = n^2$. Der Induktionsanfang ist klar, bleibt noch der Induktionsschluß zu zeigen, wir schließen von $(n-1) \rightarrow n$:

$$\begin{aligned} T(n) &= 2T(n-1) - T(n-2) + 2 \\ &= 2(n-1)^2 - (n-2)^2 + 2 \text{ (nach Induktionsannahme)} \\ &= 2n^2 - 4n + 2 - n^2 + 4n - 4 + 2 = n^2. \end{aligned}$$

(b)

Mehrmaliges Ersetzen liefert folgende Summe:

$$T(n) = n+p + T(n-1) = n+p + (n-1)+p + \ldots + 2+p + 1+p + p.$$

Ersetzt man das letzte $p$ durch $0+p$ und addiert $n-k+p + k+p = n+2p$ für $k = 0..n$, so erhält man $T(n) = (n+1)/2 \cdot (n+2p)$. Der Induktionsanfang ist klar, bleibt noch der Induktionsschluß zu zeigen, wir schließen von $(n-1) \rightarrow n$:

$$\begin{aligned} T(n) &= T(n-1) + n + p \\ &= n/2 \cdot (2p + n-1) + n + p \text{ (nach Induktionsannahme)} \\ &= n/2 \cdot (2p + n) + n/2 + p \\ &= (n+1)/2 \cdot (2p + n) - (2p + n)/2 + n/2 + p = (n+1)/2 \cdot (2p + n). \end{aligned}$$

(c)

$T(n)$ läßt sich folgendermaßen umschreiben:

$$T(n) = T(n-1) - 2[T(n-1) - T(n-2)] - 2n + 4.$$

Falls es eine Gesetzmäßigkeit zwischen aufeinanderfolgenden Werten gäbe, ließe sich $T(n)$ vereinfachen. Wir betrachten die Werte und die Differenz benachbarter Werte bis n = 3:

$T(0) = 0$,
$T(1) = 2$, $T(1)-T(0) = 2 = 2 \cdot 1$,
$T(2) = 6$, $T(2)-T(1) = 4 = 2 \cdot 2$,
$T(3) = 12$, $T(3)-T(2) = 6 = 2 \cdot 3$.

Daraus kann man $T(n) - T(n-1) = 2n$ bzw. $T(n-1) - T(n-2) = 2n-2$ vermuten. Dies führt zur vereinfachten Rekursionsformel:

$T(n) = T(n-1) + 2n = 2n + \ldots + 2 = n/2 \cdot (2n+2) = n(n+1)$.

Es bleibt noch die Gültigkeit für alle $n$ mittels vollständiger Induktion zu zeigen. Der Induktionsanfang ist klar, und wir schließen wieder von $(n-1) \rightarrow n$:

$$
\begin{aligned}
T(n) \quad & = 3T(n-1) - 2T(n-2) - 2n + 4 \\
& = 3(n-1)n - 2(n-1)(n-2) - 2n + 4 \text{ (nach Induktionsannahme)} \\
& = (n-1)\,(3n-2n+4) - 2n + 4 \\
& = (n-1)\,(n+4) - 2n + 4 \\
& = n^2 + 3n - 4 - 2n + 4 = n^2 + n = n(n+1).
\end{aligned}
$$

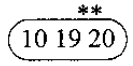
## Aufgabe 2

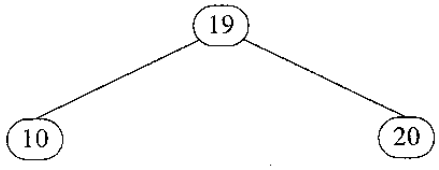
(a)

Knoten, bei denen eine Overflow-Operation durchgeführt wird, sind mit ** markiert.

Einfügen von 10, 20: kein Overflow

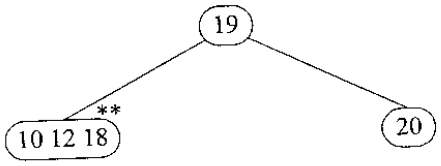
Einfügen von 19:

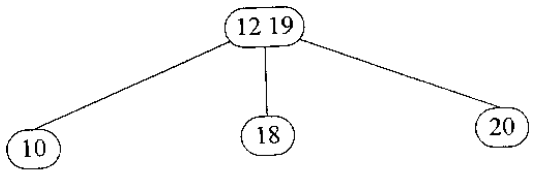



Einfügen von 18: kein Overflow

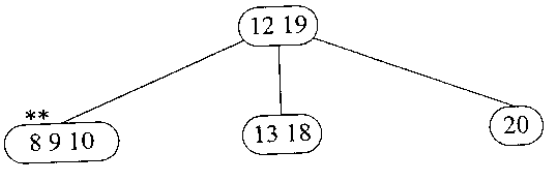
Einfügen von 12:

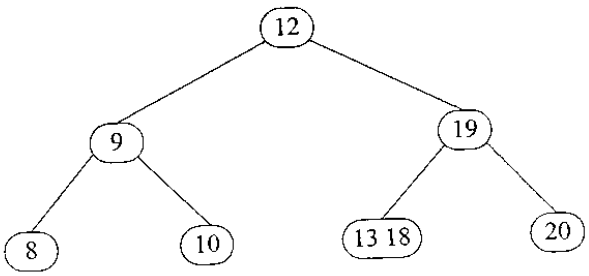



Einfügen von 13, 8: kein Overflow

Einfügen von 9:



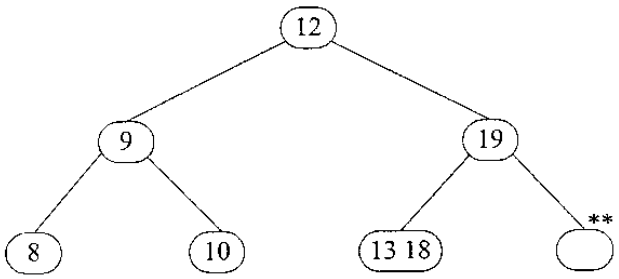



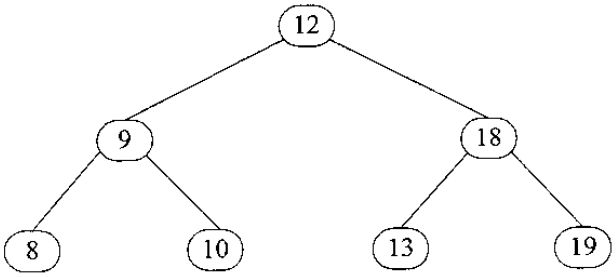

Dies ist auch gleichzeitig der Ergebnisbaum.

(b)

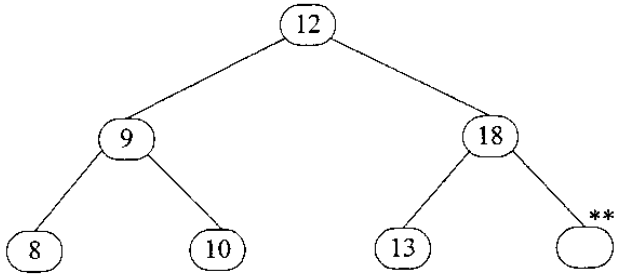
Knoten, bei denen eine Underflow-Operation durchgeführt wird, sind mit ** markiert.

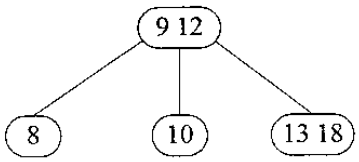
Löschen von 20 (Balance):



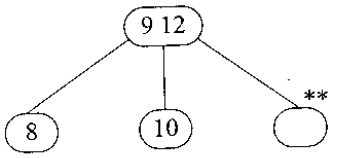

Löschen von 19 (merge):

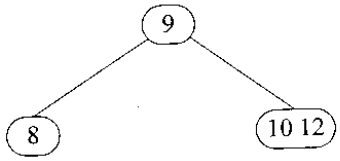



Löschen von 18: keine Underflow-Operation

Löschen von 13 (merge):




## Aufgabe 3

(a)

$h = 5$:

6, 35, 12, 22, 2, 13, 9, 28, 17

6, 9, 12, 22, 2, 13, 35, 28, 17

6, 9, 12, 17, 2, 13, 35, 28, 22 $\Rightarrow$ 5-sortierte Folge

$h = 3$:

6, 9, 12, 17, 2, 13, 35, 28, 22

6, 2, 12, 17, 9, 13, 35, 28, 22 $\Rightarrow$ 3-sortierte Folge

$h = 1$:

6, 2, 12, 17, 9, 13, 35, 28, 22

2, 6, 12, 17, 9, 13, 35, 28, 22

2, 6, 12, 9, 17, 13, 35, 28, 22

2, 6, 9, 12, 17, 13, 35, 28, 22

2, 6, 9, 12, 13, 17, 35, 28, 22

2, 6, 9, 12, 13, 17, 28, 35, 22

2, 6, 9, 12, 13, 17, 28, 22, 35

2, 6, 9, 12, 13, 17, 22, 28, 35 $\Rightarrow$ 1-sortierte Folge

(b)

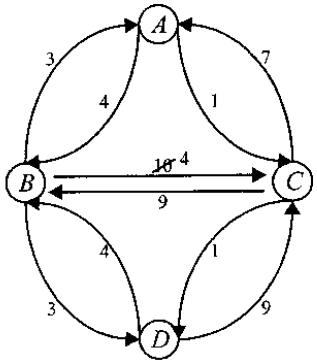
Bei dem folgenden Algorithmus gehen wir davon aus, daß die als Parameter übergebene Inkrementfolge *incr* passend zur Ausgangsfolge *a* gewählt wurde, so daß insbesondere $m \leq n$ gilt.

```
algorithm shellsort (a: array[1..n] of integer; incr: array[1..m] of integer);
var: i, j, k, h: integer;
begin
    for k := 1 to m do
        h := incr[k];
        for i := h + 1 to n do
            j := i – h;
            while j > 0 do
                if a[j] > a[j + h] then
                    vertausche(a[j], a[j + h]);
                    j := j – h;
                else j := 0
                fi
            od
        od
    od
    return a
end shellsort;
```

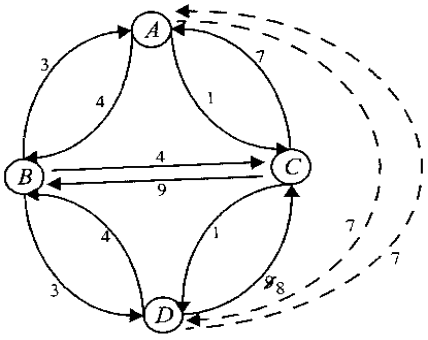
## Aufgabe 4

Die Knoten werden in lexikographische Reihenfolge bearbeitet.

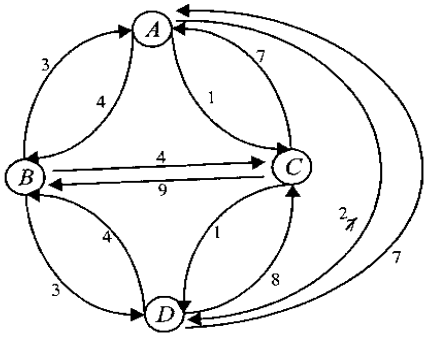
Knoten *A*:



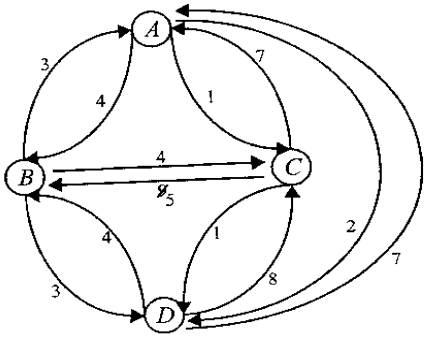
Knoten $B$:




Knoten $C$:




Knoten $D$:




Die Bearbeitung aller Knoten ist nun abgeschlossen.

## Aufgabe 5

(a)

Im ersten Schritt wird aus der Eingabemenge, die Sweep–Event–Struktur $S$ erzeugt. Haltepunkte sind die $x$–Koordinaten, an denen die Sweepline den linken oder rechten Endpunkt eines horizontalen Segmentes schneidet, oder ein vertikales Segment überlappt. Dies sind die beiden $x$–Koordinaten eines horizontalen Segmentes und die $x$–Koordinate eines vertikalen Segmentes. Jede dieser Koordinaten wird mit dem zugehörigen Segment in die Struktur eingetragen. $S$ wird nach der $x$–Koordinate sortiert.

Als Sweepline–Status–Struktur wird nun ein Range–Baum $R$ benutzt. In diesen werden horizontale Segmente anhand ihrer $y$–Koordinaten eingetragen. Zu Anfang ist der Baum leer, dann wird $S$ durchlaufen, folgende Fallunterscheidungen sind zu treffen:

(i) Das Objekt aus $S$ repräsentiert den linken Endpunkt eines Segmentes, dann füge es in $R$ ein.

(ii) Das Objekt aus $S$ repräsentiert den rechten Endpunkt eines Segmentes, dann entferne es aus $R$.

(iii) Das Objekt aus $S$ repräsentiert ein vertikales Segment, so führe auf $R$ eine Suche über das y–Intervall des Segmentes aus, die gefundenen Segmente schneiden das vertikale Segment.

$S$ enthält $O(n)$ Elemente, damit wird für das Sortieren $O(n \log n)$ Zeit benötigt. Im zweiten Schritt werden $O(n)$ Einfüge– oder Entferne–Operationen mit Laufzeit $O(\log n)$ durchgeführt. Jede Suche kostet bei $t_i$ gefundenen Elementen $O(\log n + t_i)$ Zeit, insgesamt bei $t = \Sigma\, t_i$ also $O(\log n + t)$ Zeit. Insgesamt erhält man damit eine Laufzeit von $O(n \log n + t)$.

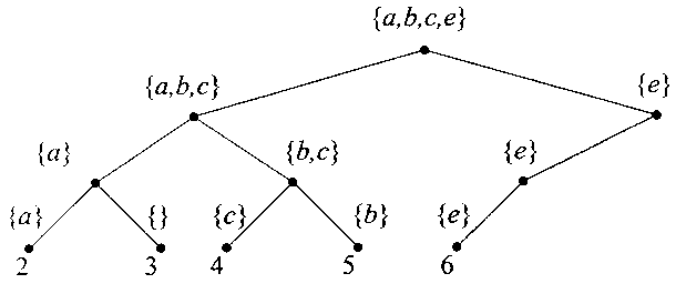
(b)

Die folgende Tabelle listet die Haltepunkte und die zugehörigen Operationen auf dem Range–Baum auf:

| # | $x$–Haltepunkt | Range–Baum Operation |
|---|---|---|
| 1 | 1 | insert $a$=2 |
| 2 | 1,5 | Query über Intervall [1,3] |
| 3 | 2 | insert $b$=5 |
| 4 | 3 | insert $c$=4 |
| 5 | 4 | insert $d$=3 |
| 6 | 5 | Query über Intervall [1, 4.5] |

| | | |
|---|---|---|
| 7 | 5,5 | insert $e$=6 |
| 8 | 6 | delete $d$ |
| 9 | 7 | Query über Intervall [3, 7] |
| 10 | 8 | delete $c$ |
| 11 | 9 | delete $b$ |
| 12 | 9,5 | delete $e$ |
| 13 | 10 | delete $a$ |

Ein Range–Baum hat den geringsten Platzbedarf, wenn er nur Knoten enthält, in denen auch Koordinaten eingetragen sind. Der Baum besitzt demnach nur 5 Blätter und könnte folgendermaßen aussehen:




Allgemein findet man die Menge aller Koordinaten, die in einem Query–Intervall $q$ enthalten sind, indem man die Wurzeln aller maximal großen Teilbäume besucht, die ganz in $q$ enthalten sind. Die Vereinigung dieser Knotenmengen ist das Ergebnis der Anfrage.

## Aufgabe 6

Der *depth–first*–Durchlauf des ungerichteten Graphen ist als *preorder*–Durchlauf (*prd*) durch den Baum zu interpretieren. Aus einem *preorder*–, und einem *postorder*–Durchlauf (*pod*) läßt sich der Baum relativ einfach rekonstruieren.

Wir wissen, daß beim *prd* zuerst der Vaterknoten genannt wird und beim *pod* der Vaterknoten immer als letzter genannt wird. $F$ ist die Wurzel. $D$ muß laut *prd* ein Sohn von $F$ sein. $C$ könnte nach *prd* nun ein Sohn von $F$ oder $D$ sein. Wenn $C$ ein Sohn von $F$ wäre, müßte $C$ aber in *pod* nach $D$ auftauchen, da $D$ auf jeden Fall der linkeste Sohn von $F$ ist. Dies ist nicht der Fall, also ist $C$ ein Sohn von $D$. $B$ muß ein weiterer Sohn von $D$ sein, da es ebenfalls in *pod* links von $D$ steht, also nicht Sohn von $F$ sein kann, und weil es rechts von $C$ steht und damit kein Sohn von

C sein kann. Mit ähnlichen Argumenten lassen sich auch alle anderen Knoten im Baum rekonstruieren. Der Ergebnisbaum sieht dann wie folgt aus: