Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

# 186.172 Algorithmen und Datenstrukturen 1 VL 4.0

## 1. Übungstest WS 2009

## 19. November 2009

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: | Vorname:

Matrikelnummer: | Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

Die Verwendung von Taschenrechnern, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Die Arbeitszeit beträgt 55 Minuten.

| | A1: | A2: | A3: | Summe: |
|---|---|---|---|---|
| Erreichbare Punkte: | 14 | 20 | 16 | 50 |
| Erreichte Punkte: | | | | |

Viel Erfolg!

**Aufgabe 1.A: $\Omega/O/\Theta$-Notation** **(14 Punkte)**

a) (8 Punkte)

Seien $f(n)$, $g(n)$ und $h(n)$ Funktionen mit positivem Wertebereich. Kreuzen Sie in der folgenden Tabelle jeweils alle richtigen Folgerungen an. Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

| Annahme | Folgerung: $f(n)$ ist in | | |
|---|---|---|---|
| | $\Omega\left(\frac{g(n)}{3}\right)$ | $O(h(n))$ | $\Theta(g(n)+h(n))$ |
| $f(n) = O(g(n)) \wedge h(n) = \Omega(g(n))$ | | | |
| $f(n) = \Omega(\sqrt{g(n)}) \wedge h(n) = \Theta(f(n))$ | | | |
| $f(n) = \Theta(g(n)) \wedge g(n) = O(h(n))$ | | | |
| $f(n) = \Theta(g(n)) \wedge h(n) = \Theta(g(n))$ | | | |

b) (6 Punkte)

Bestimmen Sie die Laufzeiten des unten angegebenen Codestücks A in Abhängigkeit von $n$ in $\Theta$-Notation, und ergänzen sie das Codestück B so, dass es in Abhängigkeit von $n$ die Laufzeit $\Theta(n^2 \log n)$ hat.

A

```
i = n^2;
j = log n;
b = 1;
while i > 0 do
  while j > 0 do
    j = j − 1;
    for k = 1000, ..., n^2 do
      b = b + 1;
    end for
  end while
  b = b/2;
  i = i − 1;
end while
```

B

```
b = 0;
for i = 1, ..., [      ] do
  j = n;
  a = 1;
  while j > 1 do
    j = [      ];
    a = 2 · a;
  end while
end for
for k = 1, ..., n^3/a do
  b = 2 · b + a;
end for
```

**Aufgabe 2.A: Sortierverfahren** **(20 Punkte)**

a) (8 Punkte)

Führen Sie auf die Zahlenfolge $\langle 7, 4, 5, 3, 2, 6, 1 \rangle$ den Algoritmus Heapsort aus um die Zahlenfolge aufsteigend zu sortieren. Geben sie das Feld nach dem Aufruf von `ErstelleHeap()` und dann nach jedem Versickerungschritt an (die graphische Darstellung des Heaps reicht *nicht*).

| | $7, 4, 5, 3, 2, 6, 1$ |
|---|---|
| nach `ErstelleHeap()` | |

b) (12 Punkte)

Gegeben sei der obige Algorithmus `sort(`$p$`)`, wobei $p$ ein Element aus einer einfach verketteten linearen Liste und $L$ ein Zeiger auf den Anfang dieser Liste ist. Der Algorithmus wird durch den Aufruf $L =$ `sort(`$L$`)` gestartet.

**Algorithmus:** `sort(`$p$`)`

```
if (p == NULL) return NULL;
if (p.next == NULL) return p;

p.next = sort(p.next);
if (p.key ≤ p.next.key) return p;

newhead = p.next;
before = p.next;
after = p.next.next;
while (after ≠ NULL ∧ after.key < p.key)
do
  before = after;
  after = after.next;
end while

before.next = p;
p.next = after;

return newhead;
```

- Auf welchem aus der Vorlesung bekannten Sortierverfahren beruht `sort()`?
- Geben Sie die Laufzeit von `sort()` im Best- und in Worst-Case in $\Theta$-Notation in Abhängigkeit der Anzahl der zu sortierenden Element $n$ an.
- Stellen Sie die Zahlenfolge $\langle 4, 3, 1, 2, 5 \rangle$ als einfach verkettete lineare Liste dar, und wenden Sie den Algorithmus `sort()` auf diese Liste an. Zeichnen Sie die Liste nach jeder Abarbeitung eines Aufrufes von `sort()`.

**Aufgabe 3.A: Abstrakte Datentypen und Suchverfahren (16 Punkte)**

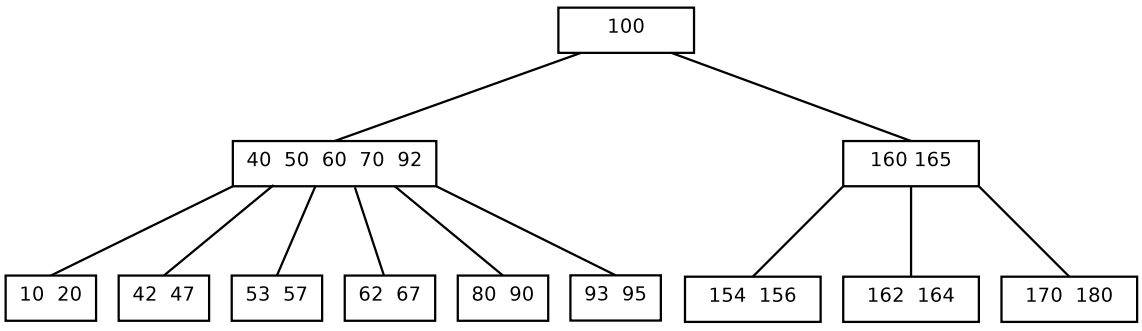
a) (11 Punkte)

Schreiben Sie einen Algorithmus in Pseudocode, um in einer *aufsteigend* sortierten, doppelt verketteten, zyklischen Liste $L$ ein Element $x$ sortiert einzufügen.

Geben Sie für diesen Algorithmus den Aufwand für den Worst-Case in $\Theta$-Notation, in Abhängigkeit der Anzahl $n$ der in $L$ gespeicherten Elemente, an.

b) (5 Punkte)

Gegeben sei untenstehender B-Baum. Geben Sie die Ordnung $m$ des B-Baumes, sowie zusätzlich vier Argumente an, warum es sich um einen gültigen B-Baum der Ordnung $m$ handelt.



Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

# 186.172 Algorithmen und Datenstrukturen 1 VL 4.0

## 1. Übungstest WS 2009

## 19. November 2009

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: | Vorname:

Matrikelnummer: | Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

Die Verwendung von Taschenrechnern, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Die Arbeitszeit beträgt 55 Minuten.

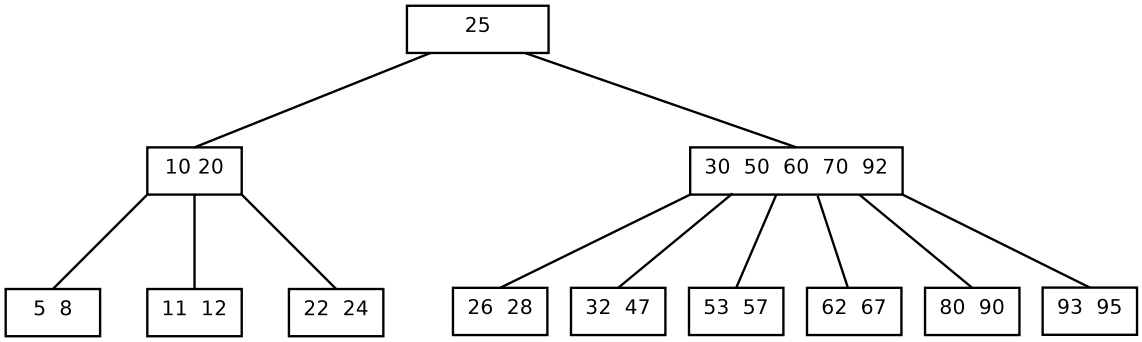
| | B1: | B2: | B3: | Summe: |
|---|---|---|---|---|
| Erreichbare Punkte: | 16 | 14 | 20 | 50 |
| Erreichte Punkte: | | | | |

Viel Glück!

**Aufgabe 1.B: Abstrakte Datentypen und Suchverfahren (16 Punkte)**

a) (5 Punkte)

Gegeben sei untenstehender B-Baum. Geben Sie die Ordnung $m$ des B-Baumes, sowie zusätzlich vier Argumente an, warum es sich um einen gültigen B-Baum der Ordnung $m$ handelt.




b) (11 Punkte)

Schreiben Sie einen Algorithmus in Pseudocode, um in einer *aufsteigend* sortierten, doppelt verketteten, zyklischen Liste $L$ ein Element $x$ sortiert einzufügen.

Geben Sie für diesen Algorithmus den Aufwand für den Worst-Case in $\Theta$-Notation, in Abhängigkeit der Anzahl $n$ der in $L$ gespeicherten Elemente, an.

**Aufgabe 2.B: $\Omega/O/\Theta$-Notation** **(14 Punkte)**

a) (6 Punkte) Ergänzen sie das Codestück A so, dass es in Abhängigkeit von $n$ die Laufzeit $\Theta(n \log n)$ hat und bestimmen Sie die Laufzeiten des unten angegebenen Codestücks B in Abhängigkeit von $n$ in $\Theta$-Notation

A

```
b = 0;
for i = 1, ..., [      ] do
  j = n;
  a = 1;
  while j > 1 do
    j = [      ];
    a = 2 · a;
  end while
end for
for k = 1, ..., n²/a do
  b = 2 · b + a;
end for
```

B

```
i = n²;
j = log n;
b = 1;
while i > 0 do
  while j > 0 do
    j = j − 1;
    for k = 1000, ..., n do
      b = b + 1;
    end for
  end while
  b = b/2;
  i = i − n;
end while
```

b) (8 Punkte)

Seien $f(n)$, $g(n)$ und $h(n)$ Funktionen mit positivem Wertebereich. Kreuzen Sie in der folgenden Tabelle jeweils alle richtigen Folgerungen an. Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

| Annahme | Folgerung: $f(n)$ ist in | | |
|---|---|---|---|
| | $\Theta(g(n) + h(n))$ | $O(h(n))$ | $\Omega\left(\frac{g(n)}{3}\right)$ |
| $f(n) = \Theta(g(n)) \wedge h(n) = \Theta(g(n))$ | | | |
| $f(n) = O(g(n)) \wedge h(n) = \Omega(g(n))$ | | | |
| $f(n) = \Theta(g(n)) \wedge g(n) = O(h(n))$ | | | |
| $f(n) = \Omega(\sqrt{g(n)}) \wedge h(n) = \Theta(f(n))$ | | | |

**Aufgabe 3.B: Sortierverfahren** **(20 Punkte)**

a) (12 Punkte)

Gegeben sei der obige Algorithmus `sort(`$p$`)`, wobei $p$ ein Element aus einer einfach verketteten linearen Liste und $L$ ein Zeiger auf den Anfang dieser Liste ist. Der Algorithmus wird durch den Aufruf $L =$ `sort(`$L$`)` gestartet.

**Algorithmus:** `sort(`$p$`)`

**if** ($p ==$ `NULL`) **return** `NULL`;
**if** ($p.next ==$ `NULL`) **return** $p$;

$p.next =$ `sort`$(p.next)$;
**if** ($p.key \leq p.next.key$) **return** $p$;

$newhead = p.next$;
$before = p.next$;
$after = p.next.next$;
**while** ($after \neq$ `NULL` $\wedge\ after.key < p.key$)
**do**
  $before = after$;
  $after = after.next$;
**end while**

$before.next = p$;
$p.next = after$;

**return** $newhead$;

- Auf welchem aus der Vorlesung bekannten Sortierverfahren beruht `sort()`?
- Geben Sie die Laufzeit von `sort()` im Best- und in Worst-Case in $\Theta$-Notation in Abhängigkeit der Anzahl der zu sortierenden Element $n$ an.
- Stellen Sie die Zahlenfolge $\langle 5, 4, 2, 3, 6\rangle$ als einfach verkettete lineare Liste dar, und wenden Sie den Algorithmus `sort()` auf diese Liste an. Zeichnen Sie die Liste nach jeder Abarbeitung eines Aufrufes von `sort()`.

b) (8 Punkte)

Führen Sie auf die Zahlenfolge $\langle 12, 9, 10, 8, 7, 11, 6\rangle$ den Algoritmus Heapsort aus um die Zahlenfolge aufsteigend zu sortieren. Geben sie das Feld nach dem Aufruf von `ErstelleHeap()` und dann nach jedem Versickerungschritt an (die graphische Darstellung des Heaps reicht *nicht*).

| | $12, 9, 10, 8, 7, 11, 6$ |
|---|---|
| nach `ErstelleHeap()` | |