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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

Nachtragstest WS 2008

28. Jänner 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:	16	16	18	50
Erreichte Punkte:				

Viel Erfolg!

Aufgabe 1.A: Ω/O/Θ-Notation **(16 Punkte)**

a) (6 Punkte) Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} n \log_3 n + 3^9 \cdot \sqrt{n}, & \text{falls } n \text{ durch 3 teilbar} \\ 9n \log n + n\sqrt{n}, & \text{sonst} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	$\Omega(\cdot)$	$\Theta(\cdot)$	$O(\cdot)$	keines
$n + \sqrt{n}$				
$n \log n + n\sqrt{n}$				
$n\sqrt[3]{n}$				

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

b) (6 Punkte) Seien $f(n)$ und $g(n)$ Funktionen mit positivem Wertebereich. Kreuzen Sie in der folgenden Tabelle an, ob die jeweilige Folgerung wahr oder falsch ist.

Aussage	wahr	falsch
$g(n) = \Omega(f(n)) \quad \Rightarrow \quad g(n) = \Omega(n \cdot f(n))$		
$g(n) = O(f(n)) \quad \Rightarrow \quad f(n) = \Theta(g(n))$		
$f(n) = O(g(n)) \wedge f(n) = \Omega(g(n)) \quad \Rightarrow \quad f(n) = \Theta(g(n))$		
$f(n) = O(4 \cdot g(n)) \quad \Rightarrow \quad g(n) = O(f(n))$		

c) (4 Punkte) Kreuzen Sie an, ob die folgenden Aussagen richtig sind oder nicht.

$\Theta(\sqrt[5]{n^2}) < \Theta(\sqrt[3]{n^2})$ ◯ ja ◯ nein

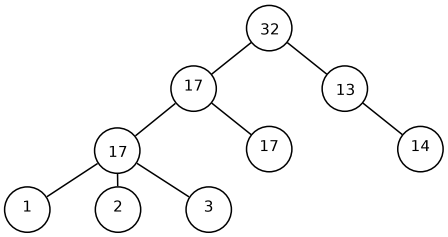
$\Theta(3 \cdot \frac{n^3}{\sqrt{n}}) > \Theta(\frac{n^3}{\sqrt{n}})$ ◯ ja ◯ nein

$\Theta(\sqrt{n}) < \Theta(\log n)$ ◯ ja ◯ nein

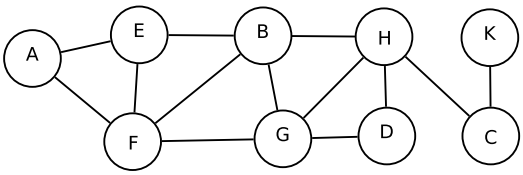
$\Theta(\log n) \geq \Theta(\log n^2)$ ◯ ja ◯ nein

Aufgabe 2.A: Allerlei **(16 Punkte)**

a) (4 Punkte) Eigentlich sollte die folgende Abbildung einen **Maximum-Heap** als Baum darstellen, allerdings haben sich einige Fehler eingeschlichen. Markieren Sie die Fehler und beschreiben Sie diese kurz (wenige Worte).



b) (6 Punkte) Gegeben sei der folgende ungerichtete Graph $G(V, E)$:



Auf diesem Graphen wird eine **Tiefensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge entstehen?

A, E, F, B, G, D, H, C, K	☐ ja ☐ nein	C, H, G, B, F, E, A, D, K	☐ ja ☐ nein
C, H, G, B, F, E, A, K, D	☐ ja ☐ nein	A, E, F, B, G, D, H, C, I	☐ ja ☐ nein

c) (6 Punkte) Vergleichen Sie die Datenstrukturen *einfach verkettete Liste*, *natürlicher binärer Suchbaum*, *sortiertes Feld* und *B-Baum* bezüglich des Aufwandes für die Suche nach dem kleinsten vorhandenen Schlüssel im Best- und Worst-Case in Θ-Notation in Abhängigkeit der Anzahl n der gespeicherten Elemente.

Datenstruktur	Best-Case	Worst-Case
verkettete Liste		
sortiertes Feld		
natürlicher binärer Suchbaum		
B-Baum		

Aufgabe 3.A: Optimierung (18 Punkte)

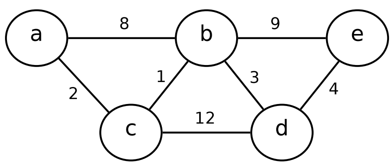
a) (8 Punkte) Betrachten Sie den folgenden Algorithmus, der als Eingabe einen zusammenhängenden Graphen $G(V, E)$ und zwei Knoten $s, t \in V$ erhält.

Foo($G(V, E), s, t$)

```
Für alle v ∈ V: Setze d[v] = ∞;
Setze Q = V und d[s] = 0;
solange Q nicht leer {
    Entnimm jenes u aus Q mit minimalem d[u];
    falls u = t dann {
        Ausgabe: d[t];
        Abbruch: Fertig;
    }
    für alle e = (u, v) mit v ∈ N(u) {
        // w_e: Gewicht der Kante e
        falls d[v] > d[u] + w_e dann {
            d[v] = d[u] + w_e;
        }
    }
}
```

Iteration	$d[a]$	$d[b]$	$d[c]$	$d[d]$	$d[e]$
Init	0	∞	∞	∞	∞
Ausgabe des Algorithmus:					

Wenden Sie den Algorithmus Foo auf den nachstehenden Graphen G an (die Kantenbeschriftungen entsprechen den Kantengewichten). Der Algorithmus wird mit Foo(G, a, e) aufgerufen. Visualisieren Sie alle Iterationsschritte, indem Sie die oben angeführte Tabelle vervollständigen. Geben Sie dabei jeweils den Zustand *nach* einer Iteration der **solange**-Schleife an.



b) (4 Punkte) Was berechnet der Algorithmus Foo und auf welchem, aus der Vorlesung bekannten, Optimierungsprinzip basiert der Algorithmus?

c) (2 Punkte) Geben Sie die Laufzeit des in Punkt (a) gegebenen Algorithmus im Worst-Case, in Abhängigkeit der Anzahl n der Knoten V eines Graphen G, in Θ-Notation an, wenn für die Verwaltung der Knoten in Q eine lineare Liste verwendet wird.

d) (4 Punkte) Erklären sie in wenigen Worten den Unterschied zwischen *dynamischer Programmierung* und *Divide and Conquer*-Verfahren.

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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

Nachtragstest WS 2008

28. Jänner 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:	18	16	16	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Optimierung (18 Punkte)

a) (4 Punkte) Erklären sie in wenigen Worten den Unterschied zwischen *dynamischer Programmierung* und *Divide and Conquer*-Verfahren.

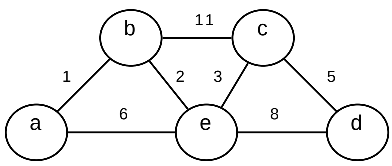
b) (8 Punkte) Betrachten Sie den folgenden Algorithmus, der als Eingabe einen zusammenhängenden Graphen $G(V, E)$ und zwei Knoten $s, t \in V$ erhält.

Foo($G(V, E), s, t$)

Für alle $v \in V$: Setze $d[v] = \infty$;
Setze $Q = V$ und $d[s] = 0$;
solange Q nicht leer **{**
  Entnimm jenes u aus Q mit minimalem $d[u]$;
  falls $u = t$ **dann {**
    Ausgabe: $d[t]$;
    Abbruch: Fertig;
  }
  für alle $e = (u, v)$ mit $v \in N(u)$ **{**
    // w_e: *Gewicht der Kante* e
    falls $d[v] > d[u] + w_e$ **dann {**
      $d[v] = d[u] + w_e$;
    }
  }
}

Iteration	$d[a]$	$d[b]$	$d[c]$	$d[d]$	$d[e]$
Init	0	∞	∞	∞	∞
Ausgabe des Algorithmus:					

Wenden Sie den Algorithmus Foo auf den nachstehenden Graphen G an (die Kantenbeschriftungen entsprechen den Kantengewichten). Der Algorithmus wird mit Foo(G, a, d) aufgerufen. Visualisieren Sie alle Iterationsschritte, indem Sie die oben angeführte Tabelle vervollständigen. Geben Sie dabei jeweils den Zustand *nach* einer Iteration der **solange**-Schleife an.



c) (4 Punkte) Was berechnet der Algorithmus Foo und auf welchem, aus der Vorlesung bekannten, Optimierungsprinzip basiert der Algorithmus?

d) (2 Punkte) Geben Sie die Laufzeit des in Punkt (b) gegebenen Algorithmus im Worst-Case, in Abhängigkeit der Anzahl n der Knoten V eines Graphen G, in Θ-Notation an, wenn für die Verwaltung der Knoten in Q eine lineare Liste verwendet wird.

Aufgabe 2.B: $\Omega/O/\Theta$-Notation **(16 Punkte)**

a) (6 Punkte) Seien $f(n)$ und $g(n)$ Funktionen mit positivem Wertebereich. Kreuzen Sie in der folgenden Tabelle an, ob die jeweilige Folgerung wahr oder falsch ist.

Aussage	wahr	falsch
$f(n) = \Theta(g(n)) \quad \Rightarrow \quad f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$		
$f(n) = \Theta(g(n)) \quad \Rightarrow \quad g(n) = O(f(n))$		
$g(n) = O(f(n)) \quad \Rightarrow \quad f(n) = O(4 \cdot g(n))$		
$g(n) = \Omega(n \cdot f(n)) \quad \Rightarrow \quad g(n) = \Omega(f(n))$		

b) (4 Punkte) Kreuzen Sie an, ob die folgenden Aussagen richtig sind oder nicht.

$\Theta(3 \cdot \frac{n^3}{\sqrt{n}}) \leq \Theta(\frac{n^3}{\sqrt{n}})$ ◯ ja ◯ nein $\quad$ $\Theta(\log n) < \Theta(\log n^2)$ ◯ ja ◯ nein

$\Theta(\sqrt[5]{n^2}) > \Theta(\sqrt[3]{n^2})$ ◯ ja ◯ nein $\quad$ $\Theta(\sqrt{n}) > \Theta(\log n)$ ◯ ja ◯ nein

c) (6 Punkte) Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} n^2 \log_2 n + 2^4 \cdot \sqrt{n}, & \text{falls } n \text{ durch 2 teilbar} \\ 4n^2 \log n + n^2\sqrt{n}, & \text{sonst} \end{cases}$$

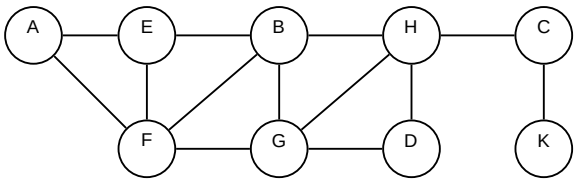
Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	$\Omega(\cdot)$	$\Theta(\cdot)$	$O(\cdot)$	keines
$n^2 \sqrt[3]{n}$				
$n^3 + \sqrt{n}$				
$n^2 \log n + n\sqrt{n}$				

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

Aufgabe 3.B: Allerlei **(16 Punkte)**

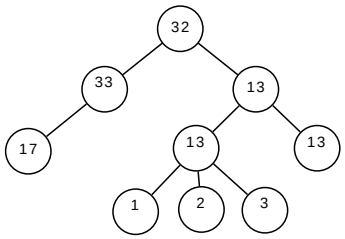
a) (6 Punkte) Gegeben sei der folgende ungerichtete Graph $G(V, E)$:



Auf diesem Graphen wird eine **Tiefensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge entstehen?

A, E, F, B, G, D, H, C, K	☐ ja ☐ nein	C, H, G, B, F, E, A, D, K	☐ ja ☐ nein
C, H, G, B, F, E, A, K, D	☐ ja ☐ nein	A, E, F, B, G, D, H, C, I	☐ ja ☐ nein

b) (4 Punkte) Eigentlich sollte die folgende Abbildung einen **Maximum-Heap** als Baum darstellen, allerdings haben sich einige Fehler eingeschlichen. Markieren Sie die Fehler und beschreiben Sie diese kurz (wenige Worte).



c) (6 Punkte) Vergleichen Sie die Datenstrukturen *einfach verkettete Liste*, *natürlicher binärer Suchbaum*, *sortiertes Feld* und *B-Baum* bezüglich des Aufwandes für die Suche nach dem größten vorhandenen Schlüssel im Best- und Worst-Case in Θ-Notation in Abhängigkeit der Anzahl n der gespeicherten Elemente.

Datenstruktur	Worst-Case	Best-Case
sortiertes Feld		
verkettete Liste		
natürlicher binärer Suchbaum		
B-Baum		