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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

Nachtragstest WS 2010

26. Januar 2011

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.

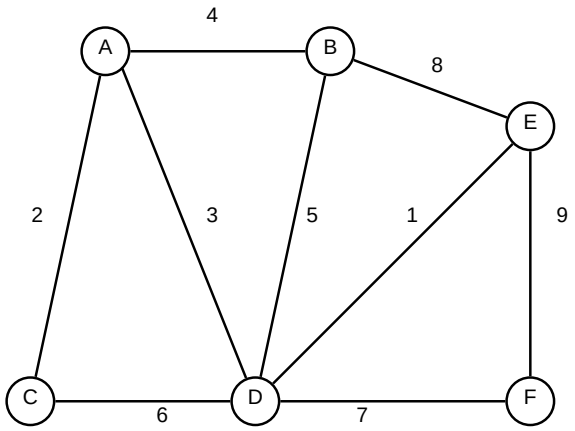
	A1:	A2:	A3:	Summe:
Erreichbare Punkte:	16	16	18	50
Erreichte Punkte:				

Viel Erfolg!

Aufgabe 1.A: Optimierung **(16 Punkte)**

a) (10 Punkte) Führen Sie anhand des nachstehend abgebildeten Graphen den Algorithmus von *Kruskal* für das Finden eines minimalen Spannbaums durch.

- Verwenden Sie zur Lösung des Problems die *Union Find* Datenstruktur und geben Sie deren Zustand nach jedem Aufruf von *union* grafisch an. Wenden Sie dabei beide in Vorlesung und Skriptum vorgestellten Verbesserungen, Methode der Pfadverkürzung(Kompressionsmethode) und Vereinigung nach Höhe, der *Union Find* Datenstruktur an.
- Markieren Sie am Ende Ihrer Berechnungen jene Kanten des Graphen in der Abbildung, die den minimalen Spannbaum bilden, und geben Sie das Gewicht des minimalen Spannbaums an.



b) (4 Punkte) Ihr Ziel der zweiten Programmieraufgabe war Heuristiken zu implementieren, die Variablenordnungen finden, für die das resultierende "Reduced Ordered Binary Decision Diagram" möglichst klein ist. Wie könnte ein exaktes Verfahren, das auf naiver Enumeration basiert, aussehen und wie viele mögliche gültige Lösungen in Abhängigkeit der Anzahl der Variablen n im Abstrakten Syntax Baum müssten evaluiert werden? Es muss kein Pseudocode angegeben werden, eine Beschreibung des Verfahren in vier bis fünf Sätzen ist ausreichend.

c) (2 Punkte) Erläutern Sie die Idee der dynamischen Programmierung.

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

(16 Punkte) Schreiben Sie in detailliertem Pseudocode eine Funktion $L_2 = sort(L_1)$, die die Elemente einer einfach verketteten azyklischen Liste L_1 aufsteigend sortiert und die sortierten Elemente in einer Liste L_2 zurück gibt.
Folgende Punkte sind dabei zu beachten:

- Die Laufzeit muss in Abhängigkeit der Anzahl der Listenelemente n durch $O(n^2)$ beschränkt sein.
- Sollte die Liste L_1 bereits auf- oder absteigend sortiert sein, dann soll die Laufzeit in Abhängigkeit der Anzahl der Listenelemente n im Worst-Case $\Theta(n)$ betragen.
- Beim Aufruf des Algorithmus gilt: $L_1 \neq NULL$.
- Es darf nur konstant viel zusätzlicher Speicherplatz verwendet werden.

Geben Sie für Ihren Algorithmus den Aufwand für den Worst-Case und Best-Case in Θ-Notation in Abhängigkeit der Anzahl der Schlüssel n, die in der Liste gespeichert sind, an.
Hinweis zur Bewertung: Für die Sonderfälle einer bereits absteigend und aufsteigend sortierten Folge werden 8 Punkte vergeben, für die sonstige Sortierung 6 Punkte und die Bestimmung der Laufzeit 2 Punkte.

Aufgabe 3.A: Diverses **(18 Punkte)**

a) (8 Punkte) Die Laufzeitanalyse eines Algorithmus hat ergeben:

$$T(n) = 10n + 10 \cdot \sum_{k=1}^{n} k$$

Beweisen oder widerlegen Sie, dass die Beziehung $T(n) = \Theta(n^2)$ gilt. Bedenken Sie, dass für einen Beweis gegebenenfalls auch geeignete Werte für die Konstanten c_1, c_2 und n_0 angegeben werden müssen.

b) (8 Punkte)

Tragen Sie die Anzahl der Schlüsselvergleiche und Schlüsselbewegungen der angegebenen Sortierverfahren (Implementierung lt. Skriptum) in Θ-Notation in Abhängigkeit der Anzahl der zu sortierenden Schlüssel n ein. Als Schlüsselbewegung wird nicht nur eine Positionsverschiebung, sondern auch das Speichern eines Schlüssels in eine temporäre Variable gewertet. Jede Spalte wird nur dann gewertet, wenn Sie vollständig richtig ist.

Schlüsselvergleiche	Selection-Sort	Merge-Sort	Heap-Sort	Insertion-Sort
Best-Case				
Worst-Case				

Schlüsselbewegungen	Selection-Sort	Merge-Sort	Heap-Sort	Insertion-Sort
Best-Case				
Worst-Case				

c) (2 Punkte)

- Welches der vier oben angeführten Sortierverfahren würden Sie bevorzugen, wenn Sie möglichst wenig zusätzlichen Speicher in Anspruch nehmen wollen? Begründen Sie Ihre Antwort.

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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

Nachtragstest WS 2010

26. Januar 2011

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: Diverses **(18 Punkte)**

a) (8 Punkte)

Tragen Sie die Anzahl der Schlüsselbewegungen und Schlüsselvergleiche der angegebenen Sortierverfahren (Implementierung lt. Skriptum) in Θ-Notation in Abhängigkeit der Anzahl der zu sortierenden Schlüssel n ein. Als Schlüsselbewegung wird nicht nur eine Positionsverschiebung, sondern auch das Speichern eines Schlüssels in eine temporäre Variable gewertet. Jede Spalte wird nur dann gewertet, wenn Sie vollständig richtig ist.

Schlüsselbewegungen	Merge-Sort	Insertion-Sort	Heap-Sort	Selection-Sort
Worst-Case				
Best-Case				

Schlüsselvergleiche	Merge-Sort	Insertion-Sort	Heap-Sort	Selection-Sort
Worst-Case				
Best-Case				

b) (2 Punkte)

- Welches der vier oben angeführten Sortierverfahren würden Sie bevorzugen, wenn Sie möglichst wenig zusätzlichen Speicher in Anspruch nehmen wollen? Begründen Sie Ihre Antwort.

c) (8 Punkte) Die Laufzeitanalyse eines Algorithmus hat ergeben:

$$T(n) = 30n + 10 \cdot \sum_{k=1}^{n} k$$

Beweisen oder widerlegen Sie, dass die Beziehung $T(n) = \Theta(n^2)$ gilt. Bedenken Sie, dass für einen Beweis gegebenenfalls auch geeignete Werte für die Konstanten c_1, c_2 und n_0 angegeben werden müssen.

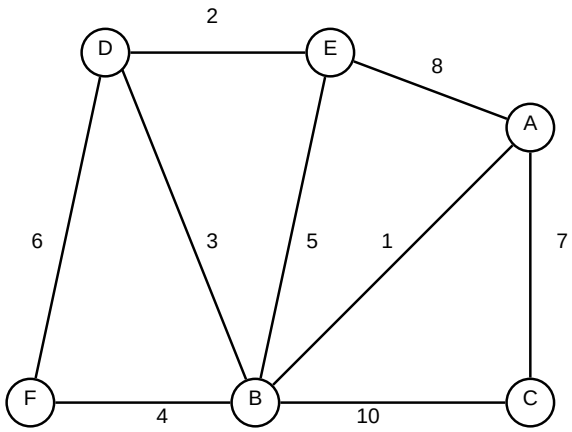
Aufgabe 2.B: Optimierung **(16 Punkte)**

a) (4 Punkte) Ihr Ziel der zweiten Programmieraufgabe war Heuristiken zu implementieren, die Variablenordnungen finden, für die das resultierende "Reduced Ordered Binary Decision Diagram" möglichst klein ist. Wie könnte ein exaktes Verfahren, das auf naiver Enumeration basiert, aussehen und wie viele mögliche gültige Lösungen in Abhängigkeit der Anzahl der Variablen n im Abstrakten Syntax Baum müssten evaluiert werden? Es muss kein Pseudocode angegeben werden, eine Beschreibung des Verfahren in vier bis fünf Sätzen ist ausreichend.

b) (2 Punkte) Erläutern Sie die Idee der dynamischen Programmierung.

c) (10 Punkte) Führen Sie anhand des nachstehend abgebildeten Graphen den Algorithmus von *Kruskal* für das Finden eines minimalen Spannbaums durch.

- Verwenden Sie zur Lösung des Problems die *Union Find* Datenstruktur und geben Sie deren Zustand nach jedem Aufruf von *union* grafisch an. Wenden Sie dabei beide in Vorlesung und Skriptum vorgestellten Verbesserungen, Methode der Pfadverkürzung(Kompressionsmethode) und Vereinigung nach Höhe, der *Union Find* Datenstruktur an.
- Markieren Sie am Ende Ihrer Berechnungen jene Kanten des Graphen in der Abbildung, die den minimalen Spannbaum bilden, und geben Sie das Gewicht des minimalen Spannbaums an.



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

(16 Punkte) Schreiben Sie in detailliertem Pseudocode eine Funktion $L_2 = sort(L_1)$, die die Elemente einer einfach verketteten azyklischen Liste L_1 aufsteigend sortiert und die sortierten Elemente in einer Liste L_2 zurück gibt.
Folgende Punkte sind dabei zu beachten:

- Die Laufzeit muss in Abhängigkeit der Anzahl der Listenelemente n durch $O(n^2)$ beschränkt sein.
- Sollte die Liste L_1 bereits auf- oder absteigend sortiert sein, dann soll die Laufzeit in Abhängigkeit der Anzahl der Listenelemente n im Worst-Case $\Theta(n)$ betragen.
- Beim Aufruf des Algorithmus gilt: $L_1 \neq NULL$.
- Es darf nur konstant viel zusätzlicher Speicherplatz verwendet werden.

Geben Sie für Ihren Algorithmus den Aufwand für den Worst-Case und Best-Case in Θ-Notation in Abhängigkeit der Anzahl der Schlüssel n, die in der Liste gespeichert sind, an.
Hinweis zur Bewertung: Für die Sonderfälle einer bereits absteigend und aufsteigend sortierten Folge werden 8 Punkte vergeben, für die sonstige Sortierung 6 Punkte und die Bestimmung der Laufzeit 2 Punkte.