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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

Nachtragstest SS 2011

30. Juni 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.

Schreiben Sie mit einem dokumentenechten, nicht roten Stift.

Die Arbeitszeit beträgt 55 Minuten.

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

Viel Erfolg!

Aufgabe 1.A: Suchen und Graphen **(20 Punkte)**

a) (12 Punkte) Schreiben Sie in detailliertem Pseudocode eine Funktion *kreissuche*(V, E), die überprüft, ob ein ungerichteter, ungewichteter Graph $G = (V, E)$ zumindest einen Kreis enthält. Der Graph ist nicht unbedingt zusammenhängend. Der Algorithmus soll auf einer **nicht-rekursiven** Variante der Tiefensuche basieren. Sie können dafür den abstrakten Datentyp *Stack* (Stapel) verwenden, der eine beliebige Anzahl an Objekten aufnehmen kann. Ein Stack `S` stellt die folgenden Operationen zur Verfügung:

- `S.isEmpty()`: Liefert *wahr* zurück, falls der Stack `S` keine Elemente enthält und *falsch* sonst.
- `S.push(`x`)`: Legt das Element x auf den Stack `S`.
- `S.pop()`: Entfernt das zuletzt eingefügte Element vom Stack `S` und liefert dieses Element zurück. Das Ergebnis ist undefiniert, wenn S leer ist.

b) (8 Punkte) Gegeben ist die Zahlenfolge

$$\langle 2, 4, 6, 8, 7, 5, 3, 1 \rangle.$$

Fügen Sie alle Schlüssel der Folge in gegebener Reihenfolge nacheinander in einen anfangs leeren B-Baum der Ordnung 5 ein. Zeichnen Sie den Baum zumindest jeweils nach dem Einfügen der Schlüssel 6, 5 und 1. (Die leeren Blätter können bei der Zeichnung entfallen.)

Aufgabe 2.A: Optimierung und 2.Programmieraufgabe **(16 Punkte)**

a) (10 Punkte) Gegeben sei folgende Instanz des 0/1-Rucksackproblems mit Kapazität $K = 4$ und folgenden Gegenständen:

Gegenstand i	Gewicht w_i	Wert c_i
1	4	4
2	2	3
3	2	2
4	3	4
5	1	1

$i\backslash j$	0	1	2	3	4
0	0	0	0	0	0
1					
2					
3					
4					
5					

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende 6×5-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher Wert mit den ersten i Gegenständen erreicht werden kann, wenn das Gesamtgewicht der gewählten Gegenstände kleiner oder gleich j ist.

Die Felder der Matrix können wie folgt berechnet werden:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 4$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\, m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 5 \\ j = 0, \ldots, 4 \end{cases}$$

- (6 Punkte) Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.
- (2 Punkte) Geben Sie die Laufzeit für das Befüllen der Matrix im Worst- und im Best-Case in Θ-Notation, in Abhängigkeit der Anzahl der Elemente n und der Kapazität K, an.
- (2 Punkte) Geben Sie die gewählten Gegenstände der gefundenen optimalen Lösung an.

b) (6 Punkte) Vervollständigen Sie folgende Aussagen über die zweite Programmieraufgabe:

Jeder Knoten des vollständigen Graphens ist zumindest ___ Mal in der Lösung enthalten.
Eine Kante des vollständigen Graphens darf höchstens ___ Mal in der Lösung enthalten sein.
Der vollständige Graph muss zumindest aus ___ Knoten bestehen, damit eine gültige Lösung gefunden werden kann.

Aufgabe 3.A: Sortierverfahren und Notationen **(14 Punkte)**

a) (6 Punkte) Sortieren Sie die folgende Zahlenfolge in aufsteigend sortierter Reihenfolge mit Hilfe von Quick-Sort (Implementierung laut Vorlesung bzw. Skriptum, Pivotelement ist jeweils das letzte Element der zu sortierenden Teilfolge)

$$\langle 5, 7, 3, 4, 8, 1 \rangle.$$

Schreiben Sie nach jedem Aufruf der Funktion `partition` des Algorithmus die entstandene Zahlenfolge auf und markieren Sie alle Elemente, die bereits ihre endgültige Position erreicht haben.

b) (8 Punkte) Gegeben Sei die Funktion

$$f(n) = \begin{cases} n^2 + 5 \cdot n, & \text{falls } n < 100 \\ 10 \cdot n^3 + n + 5, & \text{sonst} \end{cases}$$

- Beweisen oder widerlegen Sie, dass für die Funktion $f(n)$ die Beziehung $f(n) = O(n^2)$ gilt. Beachten Sie, dass auch die formale Definition der O-Notation angegeben werden muss.
- Kreuzen Sie anschließend in der folgenden Tabelle die zutreffenden Felder für die oben angeführte Funktion $f(n)$ an:

$f(n)$ ist	$\Theta(.)$	$O(.)$	$\Omega(.)$	keines
$n^2\sqrt{n}$				
n^3				
n^2				
$n^2 \log n$				

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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

Nachtragstest SS 2011

30. Juni 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.

Schreiben Sie mit einem dokumentenechten, nicht roten Stift.

Die Arbeitszeit beträgt 55 Minuten.

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

Viel Glück!

Aufgabe 1.B: Sortierverfahren und Notationen **(14 Punkte)**

a) (8 Punkte) Gegeben Sei die Funktion

$$f(n) = \begin{cases} n^3 + 5 \cdot n, & \text{falls } n > 50 \\ 10 \cdot n^2 + n + 5, & \text{sonst} \end{cases}$$

- Beweisen oder widerlegen Sie, dass für die Funktion $f(n)$ die Beziehung $f(n) = O(n^2)$ gilt. Beachten Sie, dass auch die formale Definition der O-Notation angegeben werden muss.
- Kreuzen Sie anschließend in der folgenden Tabelle die zutreffenden Felder für die oben angeführte Funktion $f(n)$ an:

$f(n)$ ist	$\Theta(.)$	$O(.)$	$\Omega(.)$	keines
$n^2 \log n$				
$n^2\sqrt{n}$				
n^3				
n^2				

b) (6 Punkte) Sortieren Sie die folgende Zahlenfolge in aufsteigend sortierter Reihenfolge mit Hilfe von Quick-Sort (Implementierung laut Vorlesung bzw. Skriptum, Pivotelement ist jeweils das letzte Element der zu sortierenden Teilfolge)

$$\langle 6, 8, 4, 5, 9, 2 \rangle.$$

Schreiben Sie nach jedem Aufruf der Funktion `partition` des Algorithmus die entstandene Zahlenfolge auf und markieren Sie alle Elemente, die bereits ihre endgültige Position erreicht haben.

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

a) (10 Punkte) Gegeben sei folgende Instanz des 0/1-Rucksackproblems mit Kapazität $K = 4$ und folgenden Gegenständen:

Gegenstand i	Gewicht w_i	Wert c_i
1	4	6
2	2	4
3	3	5
4	2	3
5	1	2

$i \backslash j$	0	1	2	3	4
0	0	0	0	0	0
1					
2					
3					
4					
5					

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende 6×5-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher Wert mit den ersten i Gegenständen erreicht werden kann, wenn das Gesamtgewicht der gewählten Gegenstände kleiner oder gleich j ist.

Die Felder der Matrix können wie folgt berechnet werden:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 4$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\, m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 5 \\ j = 0, \ldots, 4 \end{cases}$$

- (6 Punkte) Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.
- (2 Punkte) Geben Sie die Laufzeit für das Befüllen der Matrix im Worst- und im Best-Case in Θ-Notation, in Abhängigkeit der Anzahl der Elemente n und der Kapazität K, an.
- (2 Punkte) Geben Sie die gewählten Gegenstände der gefundenen optimalen Lösung an.

b) (6 Punkte) Vervollständigen Sie folgende Aussagen über die zweite Programmieraufgabe:

Eine Kante des vollständigen Graphens darf höchstens ___ Mal in der Lösung enthalten sein.
Jeder Knoten des vollständigen Graphens ist zumindest ___ Mal in der Lösung enthalten.
Der vollständige Graph muss zumindest aus ___ Knoten bestehen, damit eine gültige Lösung gefunden werden kann.

Aufgabe 3.B: Suchen und Graphen **(20 Punkte)**

a) (12 Punkte) Schreiben Sie in detailliertem Pseudocode eine Funktion *kreissuche*(V, E), die überprüft, ob ein ungerichteter, ungewichteter Graph $G = (V, E)$ zumindest einen Kreis enthält. Der Graph ist nicht unbedingt zusammenhängend. Der Algorithmus soll auf einer **nicht-rekursiven** Variante der Tiefensuche basieren. Sie können dafür den abstrakten Datentyp *Stack* (Stapel) verwenden, der eine beliebige Anzahl an Objekten aufnehmen kann. Ein Stack `S` stellt die folgenden Operationen zur Verfügung:

- `S.isEmpty()`: Liefert *wahr* zurück, falls der Stack `S` keine Elemente enthält und *falsch* sonst.
- `S.push(`x`)`: Legt das Element x auf den Stack `S`.
- `S.pop()`: Entfernt das zuletzt eingefügte Element vom Stack `S` und liefert dieses Element zurück. Das Ergebnis ist undefiniert, wenn S leer ist.

b) (8 Punkte) Gegeben ist die Zahlenfolge

$$\langle 4, 6, 2, 1, 3, 5, 8, 7 \rangle.$$

Fügen Sie alle Schlüssel der Folge in gegebener Reihenfolge nacheinander in einen anfangs leeren B-Baum der Ordnung 5 ein. Zeichnen Sie den Baum zumindest jeweils nach dem Einfügen der Schlüssel 2, 5 und 7. (Die leeren Blätter können bei der Zeichnung entfallen.)