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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

2. Übungstest WS 2010

14. 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)**

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	10
2	2	6
3	2	5
4	1	1
5	3	10

$i \backslash j$	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:

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

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

a) (6 Punkte) Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.

b) (8 Punkte) Schreiben Sie in detailiertem Pseudocode eine Funktion *reconstruct*(m, n, K, c, w), die alle gewählten Gegenstände der Lösung ausgibt, wobei m eine Matrix, n die Anzahl der Gegenstände, K das Maximalgewicht, c der Wert der Gegenstände und w das Gewicht der Gegenstände ist. Auf den Wert sowie auf das Gewicht des i-ten Gegenstandes kann mit $c[i]$ bzw $w[i]$ zugegriffen werden. Auf den Eintrag im Feld $m_{i,j}$ kann mit $m[i][j]$ zugegriffen werden.Die Berechnung der Matrix wurde mit oben angeführten Beschreibung durchgeführt. Es soll die optimale Lösung mit minimalem Gewicht ausgegeben werden.

c) (2 Punkte) Welche Laufzeit ergibt sich bei Ihrem Pseudocode im Worst- und im Best-Case in Θ-Notation in Abhängigkeit der Anzahl der Elemente n und der Kapazität K?

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

a) (8 Punkte)

Gegeben sei eine Hashtabelle mit Tabellengröße $m = 7$. Fügen Sie nun die Zahlen

$$\langle 10, 7, 5, 19, 15, 14 \rangle$$

in gegebener Reihenfolge in eine Anfangs leere Hashtabelle ein. Als Hashfunktion soll $h(k) = (k \ mod \ 5) + 1$ und zur Kollisionsbehandlung Verkettung der Überläufer verwendet werden.

- Zeichnen Sie die resultierende Hashtabelle. Die notwendigen Hashwerte können Sie der folgenden Tabelle entnehmen.

Hashfunktion \ Schlüssel k	10	7	5	19	15	14
$h(k)$	1	3	1	5	1	5

- Ist die oben angeführte Hashfunktion $h(k)$ für die Tabellengröße 7 eine gute Wahl? Begründen Sie ihre Antwort.

b) (8 Punkte)

Gegeben seien zwei **unabhängige** Hashtabellen mit Tabellengröße $m = 7$ in denen bereits Schlüssel eingefügt wurden. Als Hashfunktionen sollen

$$h_1(k) = (k \ mod \ 5) + 1$$
$$h_2(k) = (k \ mod \ 7)$$

und zur Kollisionsbehandlung Double-Hashing mit der Verbesserung nach Brent verwendet werden.

- Fügen Sie in die folgende Hashtabelle den Schlüssel 11 ein.

Schlüssel \ Index	0	1	2	3	4	5	6
k			6	12	8	14	

- Fügen Sie in die folgende Hashtabelle den Schlüssel 15 ein.

Schlüssel \ Index	0	1	2	3	4	5	6
k	3	10	16	12	8		

- Sind die oben angeführten Hashfunktionen $h_1(k)$ bzw. $h_2(k)$ für eine beliebige Hashtabelle der Größe 7 eine gute Wahl? Begründen Sie ihre Antwort.

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

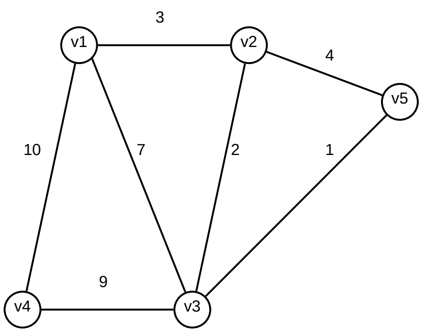
Gegeben sei folgender Algorithmus *WasBinIch*, der auf einen ungerichteten, zusammenhängenden, gewichteten Graphen $G(V,E)$ angewendet wird. Der Parameter v ist ein Knoten aus dem Graphen G.

Algorithmus WasBinIch$(G(V,E), v)$

```
Globale Variable: G(V, E);
Globale Variable: Feld previous;
found = true;
solange found == true {
   für alle w ∈ V {
      previous[w] = NULL;
   }
   found = FUNKTION1(v);
}
retourniere G(V, E);
```

Algorithmus FUNKTION1(v)

```
S = neuer Stack;
previous[v] = v;
S.push(v); // fügt Element vorne hinzu
solange nicht S.isEmpty() {
   k = S.pop();     // entfernt vorderstes
                    // Element und liefert es zurück
   für alle Knoten w ∈ N(k) {
      falls previous[w] == NULL dann
      {
         S.push(w);
         previous[w] = k;
      } sonst falls previous[k] ≠ w dann
      {
         E = E\{(k, w)};
         return true;
      }
   }
}
return false;
```



Graph G_1

a) (8 Punkte) Wenden Sie den Algorithmus *WasBinIch* durch den Aufruf $G_2 = \text{WasBinIch}(G_1(V,E), v4)$ auf den gegebenen Graphen G_1 an und zeichnen Sie den Graphen G_2. Das Auslesen der Nachbarn $w \in N(k)$ eines Knoten k erfolgt, bezogen auf die Knotenbezeichnung, in lexikographischer Reihenfolge.

b) (4 Punkte)

- Auf welchem aus der Vorlesung bekannten Verfahren beruht *WasBinIch*?
- Was berechnet der Algorithmus *WasBinIch*?

c) (6 Punkte) Kreuzen Sie zutreffende Aussagen an. Jede Zeile wird nur dann gewertet wenn Sie vollständig richtig ist.

- Die Laufzeit (Worst-Case) beträgt bei einem vollständigen Graph:
  $\Theta(|V|^2)$ ☐ $\Theta(|V|^2 log|V|)$ ☐ $\Theta(|V|^3)$ ☐ keine der angeführten ☐

- Die Laufzeit (Worst-Case) beträgt bei einem dünnen Graphen ($|E| = \Theta(|V|)$):
  $\Theta(|V|^2)$ ☐ $\Theta(|V|^2 log|V|)$ ☐ $\Theta(|V|^3)$ ☐ keine der angeführten ☐

- Bei dem oben angeführten Algorithmus handelt es sich um ein...
  rekursives Programm ☐ iteratives Programm ☐ keines von beiden ☐

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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

2. Übungstest WS 2010

14. 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:	16	16	18	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Hashverfahren **(16 Punkte)**

a) (8 Punkte)

Gegeben seien zwei **unabhängige** Hashtabellen mit Tabellengröße $m = 7$ in denen bereits Schlüssel eingefügt wurden. Als Hashfunktionen sollen

$$h_1(k) = (k \ mod \ 5) + 1$$
$$h_2(k) = (k \ mod \ 7)$$

und zur Kollisionsbehandlung Double-Hashing mit der Verbesserung nach Brent verwendet werden.

- Fügen Sie in die folgende Hashtabelle den Schlüssel 15 ein.

Schlüssel \ Index	0	1	2	3	4	5	6
k		10	1	12	8		

- Fügen Sie in die folgende Hashtabelle den Schlüssel 8 ein.

Schlüssel \ Index	0	1	2	3	4	5	6
k		5		12	10	14	

- Sind die oben angeführten Hashfunktionen $h_1(k)$ bzw. $h_2(k)$ für eine beliebige Hashtabelle der Größe 7 eine gute Wahl? Begründen Sie ihre Antwort.

b) (8 Punkte)

Gegeben sei eine Hashtabelle mit Tabellengröße $m = 7$. Fügen Sie nun die Zahlen

$$\langle 5, 12, 10, 14, 15, 19 \rangle$$

in gegebener Reihenfolge in eine Anfangs leere Hashtabelle ein. Als Hashfunktion soll $h(k) = (k \ mod \ 5) + 1$ und zur Kollisionsbehandlung Verkettung der Überläufer verwendet werden.

- Zeichnen Sie die resultierende Hashtabelle. Die notwendigen Hashwerte können Sie der folgenden Tabelle entnehmen.

Hashfunktion \ Schlüssel k	5	12	10	14	15	19
$h(k)$	1	3	1	5	1	5

- Ist die oben angeführte Hashfunktion $h(k)$ für die Tabellengröße 7 eine gute Wahl? Begründen Sie ihre Antwort.

Aufgabe 2.B: Optimierung **(16 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	2	6
2	4	11
3	2	7
4	3	11
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:

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

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

a) (6 Punkte) Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.

b) (8 Punkte) Schreiben Sie in detailiertem Pseudocode eine Funktion $reconstruct(m, n, K, c, w)$, die alle gewählten Gegenstände der Lösung ausgibt, wobei m eine Matrix, n die Anzahl der Gegenstände, K das Maximalgewicht, c der Wert der Gegenstände und w das Gewicht der Gegenstände ist. Auf den Wert sowie auf das Gewicht des i-ten Gegenstandes kann mit $c[i]$ bzw $w[i]$ zugegriffen werden. Auf den Eintrag im Feld $m_{i,j}$ kann mit $m[i][j]$ zugegriffen werden.Die Berechnung der Matrix wurde mit oben angeführten Beschreibung durchgeführt. Es soll die optimale Lösung mit minimalem Gewicht ausgegeben werden.

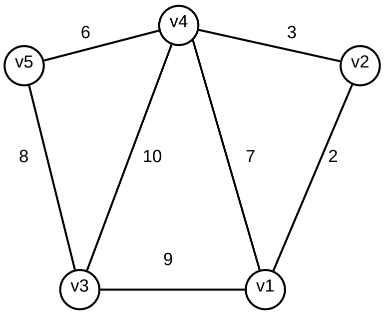
c) (2 Punkte) Welche Laufzeit ergibt sich bei Ihrem Pseudocode im Worst- und im Best-Case in Θ-Notation in Abhängigkeit der Anzahl der Elemente n und der Kapazität K?

Aufgabe 3.B: Graphen **(18 Punkte)**

Gegeben sei folgender Algorithmus *WasBinIch*, der auf einen ungerichteten, zusammenhängenden, gewichteten Graphen $G(V, E)$ angewendet wird. Der Parameter v ist ein Knoten aus dem Graphen G.

Algorithmus WasBinIch($G(V,E), v$)

```
Globale Variable: G(V, E);
Globale Variable: Feld previous;
found = true;
solange found == true {
   für alle w ∈ V {
      previous[w] = NULL;
   }
   found = FUNKTION1(v);
}
retourniere G(V, E);
```



Graph G_1

Algorithmus FUNKTION1(v)

```
S = neuer Stack;
previous[v] = v;
S.push(v); // fügt Element vorne hinzu
solange nicht S.isEmpty() {
   k = S.pop();      // entfernt vorderstes
                     // Element und liefert es zurück
   für alle Knoten w ∈ N(k) {
      falls previous[w] == NULL dann
      {
         S.push(w);
         previous[w] = k;
      } sonst falls previous[k] ≠ w dann
      {
         E = E\{(k, w)};
         return true;
      }
   }
}
return false;
```

a) (8 Punkte) Wenden Sie den Algorithmus *WasBinIch* durch den Aufruf G_2 = WasBinIch($G_1(V,E), v5$) auf den gegebenen Graphen G_1 an und zeichnen Sie den Graphen G_2. Das Auslesen der Nachbarn $w \in N(k)$ eines Knoten k erfolgt, bezogen auf die Knotenbezeichnung, in lexikographischer Reihenfolge.

b) (4 Punkte)

- Auf welchem aus der Vorlesung bekannten Verfahren beruht *WasBinIch*?
- Was berechnet der Algorithmus *WasBinIch*?

c) (6 Punkte) Kreuzen Sie zutreffende Aussagen an. Jede Zeile wird nur dann gewertet wenn Sie vollständig richtig ist.

- Die Laufzeit (Worst-Case) beträgt bei einem dünnen Graphen ($|E| = \Theta(|V|)$):
  $\Theta(|V|^2)$ ☐ $\Theta(|V|^2 log|V|)$ ☐ $\Theta(|V|^3)$ ☐ keine der angeführten ☐

- Die Laufzeit (Worst-Case) beträgt bei einem vollständigen Graph:
  $\Theta(|V|^2)$ ☐ $\Theta(|V|^2 log|V|)$ ☐ $\Theta(|V|^3)$ ☐ keine der angeführten ☐

- Bei dem oben angeführten Algorithmus handelt es sich um ein...
  rekursives Programm ☐ iteratives Programm ☐ keines von beiden ☐