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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

2. Übungstest SS 2010

11. Juni 2010

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	22	12	50
Erreichte Punkte:				

Viel Erfolg!

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

a) (12 Punkte)

Gegeben seien drei unterschiedliche Hashtabellen mit Tabellengröße $m = 7$, die zur Kollisionsbehandlung Double Hashing verwenden. Fügen Sie die angegebenen Werte jeweils in die dafür vorgesehenen Tabellen ein. Verwenden Sie hierfür die folgenden Hashfunktionen:

$$h_1(k) = k \bmod 7$$
$$h_2(k) = (k \bmod 6) + 2$$

Wird ein bereits eingefügtes Element verschoben, so muss die neue Position dieses Elementes eindeutig gekennzeichnet werden.

- Fügen Sie den Wert 10 **mit** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
	8	2	3			

- Fügen Sie den Wert 8 **mit** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
	1		10		12	

- Fügen Sie den Wert 9 **ohne** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
7		2				

- Welches Problem kann beim Einfügen in die Hashtabelle bei der Verwendung der oben angegebenen Hashfunktionen auftreten? Wie müssen die Funktionen $h_1(k)$ und/oder $h_2(k)$ abgeändert werden, um dieses Problem zu beheben?

b) (4 Punkte)

Beschreiben Sie kurz, was mit primären Häufungen in Bezug auf offene Hashverfahren gemeint ist und geben Sie an, wodurch diese bei Double Hashing verhindert werden.

Aufgabe 2.A: Graphen **(22 Punkte)**

a) (16 Punkte)

Für die Datenstruktur *Union Find* benötigt man ein Array *father*, das jedem Knoten v einen Vorgänger $father[v]$ zuweist, sowie die Prozeduren **makeset**(v), **findset**(v) und **union**(v, w). Um die in der Vorlesung behandelte Verbesserung *Vereinigung nach Höhe* zu realisieren, braucht man zusätzlich ein Array *height*, das zu jedem Knoten v die Höhe $height[v]$ des entsprechenden Unterbaums speichert.

Die Prozedur **makeset**(v) ist dabei wie folgt implementiert:

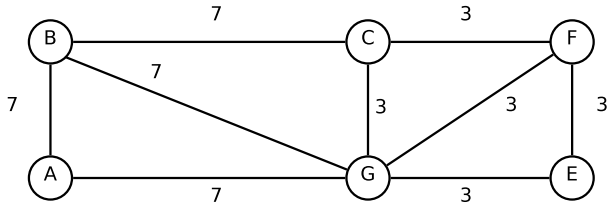
Algorithmus 1: makeset(v)

1: $father[v] = v$;
2: $height[v] = 0$;

- Schreiben Sie detaillierten Pseudocode für die Prozedur **union**(v, w) (v und w sind jeweils die Repräsentanten ihrer Menge), der die Verbesserung *Vereinigung nach Höhe* realisiert.
- Schreiben Sie detaillierten Pseudocode für die Prozedur **findset**(v) (v ist ein beliebiger Knoten des Graphen), der die Verbesserung *Pfadverkürzung (Kompressionsmethode)* realisiert. Die Aktualisierung des Arrays *height* muss in der Prozedur **findset**(v) nicht berücksichtigt werden.
- Für welchen aus der Vorlesung bekannten Algorithmus ist die *Union Find* Datenstruktur eine effiziente Möglichkeit, Kreise zu finden?

b) (6 Punkte)

Gegeben ist der folgende gewichtete ungerichtete Graph G:



- Führen Sie im Graphen G die Algorithmen von *Prim* und *Kruskal* zum Finden eines minimalen Spannbaums durch (die Kantengewichte stehen bei den Kanten) und tragen Sie *jeweils* das Gesamtgewicht des minimalen Spannbaums in die dafür vorgesehene Tabelle ein. *Falls* Sie einen Startknoten benötigen, verwenden Sie den Knoten B. Die Auswahl der Kanten muss nicht angegeben werden.

	Kruskal	Prim
Gesamtgewicht		

- Gegeben sei ein vollständiger ungerichteter gewichteter Graph G mit n Knoten, wobei alle Kanten das gleiche Gewicht g aufweisen. Geben Sie das Gesamtgewicht des minimalen Spannbaums an, wenn für die Berechnung der Algorithmus von *Kruskal* verwendet wird.

Aufgabe 3.A: Optimierung **(12 Punkte)**

a) (12 Punkte)

Gegeben sei folgende Instanz des 0/1-Rucksackproblems mit Kapazität $K = 5$ und folgenden Gegenständen:

Gegenstand i	Gewicht w_i	Wert c_i
1	2	4
2	2	5
3	2	2
4	1	2
5	3	5
6	1	4
7	6	12

$i\backslash j$	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						
4						
5						
6						
7						

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende 8×6-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher maximale 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:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 5$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\ m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 7 \\ j = 0, \ldots, 5 \end{cases}$$

- Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.

- Geben Sie eine optimale Lösung der Probleminstanz an, und markieren Sie in der Matrix jene Zellen, die beim Rekonstruieren der Lösung betrachtet werden.

- Welche Laufzeit ergibt sich bei dieser Rekonstruktion im Worst-Case in Θ-Notation in Abhängigkeit der Anzahl der Gegenstände n?

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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

2. Übungstest SS 2010

11. Juni 2010

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:	12	22	16	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Optimierung **(12 Punkte)**

a) (12 Punkte)

Gegeben sei folgende Instanz des 0/1-Rucksackproblems mit Kapazität $K = 5$ und folgenden Gegenständen:

Gegenstand i	Gewicht w_i	Wert c_i
1	1	1
2	1	3
3	2	3
4	2	5
5	2	1
6	3	4
7	6	13

$i \backslash j$	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						
4						
5						
6						
7						

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende 8×6-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher maximale 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:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 5$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\ m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 7 \\ j = 0, \ldots, 5 \end{cases}$$

- Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.

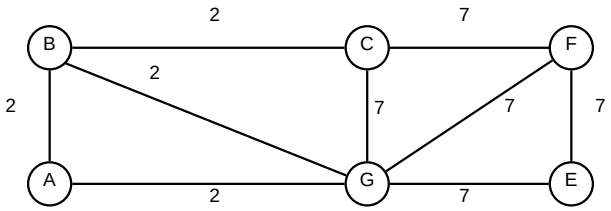
- Geben Sie eine optimale Lösung der Probleminstanz an, und markieren Sie in der Matrix jene Zellen, die beim Rekonstruieren der Lösung betrachtet werden.

- Welche Laufzeit ergibt sich bei dieser Rekonstruktion im Worst-Case in Θ-Notation in Abhängigkeit der Anzahl der Gegenstände n?

Aufgabe 2.B: Graphen **(22 Punkte)**

a) (6 Punkte)

Gegeben ist der folgende gewichtete ungerichtete Graph G:



- Führen Sie im Graphen G die Algorithmen von *Prim* und *Kruskal* zum Finden eines minimalen Spannbaums durch (die Kantengewichte stehen bei den Kanten) und tragen Sie *jeweils* das Gesamtgewicht des minimalen Spannbaums in die dafür vorgesehene Tabelle ein. *Falls* Sie einen Startknoten benötigen, verwenden Sie den Knoten B. Die Auswahl der Kanten muss nicht angegeben werden.

	Kruskal	Prim
Gesamtgewicht		

- Gegeben sei ein vollständiger ungerichteter gewichteter Graph G mit n Knoten, wobei alle Kanten das gleiche Gewicht g aufweisen. Geben Sie das Gesamtgewicht des minimalen Spannbaums an, wenn für die Berechnung der Algorithmus von *Kruskal* verwendet wird.

b) (16 Punkte)

Für die Datenstruktur *Union Find* benötigt man ein Array *father*, das jedem Knoten v einen Vorgänger $father[v]$ zuweist, sowie die Prozeduren **makeset**(v), **findset**(v) und **union**(v, w). Um die in der Vorlesung behandelte Verbesserung *Vereinigung nach Höhe* zu realisieren, braucht man zusätzlich ein Array *height*, das zu jedem Knoten v die Höhe $height[v]$ des entsprechenden Unterbaums speichert.

Die Prozedur **makeset**(v) ist dabei wie folgt implementiert:

Algorithmus 1: makeset(v)

1: $father[v] = v$;
2: $height[v] = 0$;

- Schreiben Sie detaillierten Pseudocode für die Prozedur **union**(v, w) (v und w sind jeweils die Repräsentanten ihrer Menge), der die Verbesserung *Vereinigung nach Höhe* realisiert.
- Schreiben Sie detaillierten Pseudocode für die Prozedur **findset**(v) (v ist ein beliebiger Knoten des Graphen), der die Verbesserung *Pfadverkürzung (Kompressionsmethode)* realisiert. Die Aktualisierung des Arrays *height* muss in der Prozedur **findset**(v) nicht berücksichtigt werden.
- Für welchen aus der Vorlesung bekannten Algorithmus ist die *Union Find* Datenstruktur eine effiziente Möglichkeit, Kreise zu finden?

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

a) (4 Punkte)

Beschreiben Sie kurz, was mit primären Häufungen in Bezug auf offene Hashverfahren gemeint ist und geben Sie an, wodurch diese bei Double Hashing verhindert werden.

b) (12 Punkte)

Gegeben seien drei unterschiedliche Hashtabellen mit Tabellengröße $m = 7$, die zur Kollisionsbehandlung Double Hashing verwenden. Fügen Sie die angegebenen Werte jeweils in die dafür vorgesehenen Tabellen ein. Verwenden Sie hierfür die folgenden Hashfunktionen:

$$h_1(k) = k \bmod 7$$
$$h_2(k) = (k \bmod 6) + 2$$

Wird ein bereits eingefügtes Element verschoben, so muss die neue Position dieses Elementes eindeutig gekennzeichnet werden.

- Fügen Sie den Wert 9 **ohne** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
14		2				

- Fügen Sie den Wert 10 **mit** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
	8	9	3			

- Fügen Sie den Wert 8 **mit** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
	1		10		5	

- Welches Problem kann beim Einfügen in die Hashtabelle bei der Verwendung der oben angegebenen Hashfunktionen auftreten? Wie müssen die Funktionen $h_1(k)$ und/oder $h_2(k)$ abgeändert werden, um dieses Problem zu beheben?