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 2011

10. 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.

Die Arbeitszeit beträgt 55 Minuten.

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

Viel Erfolg!

Aufgabe 1.A: Hashverfahren **(18 Punkte)**

a) (6 Punkte)

Gegeben seien zwei 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:

$$h_1(k) = k \bmod 7$$
$$h_2(k) = (k \bmod 5) + 3$$

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

- Fügen Sie den Wert 7 **mit** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
21			3	4	5	

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

0	1	2	3	4	5	6
			3			6

b) (12 Punkte) Gegeben ist eine Hashtabelle in Form eines Feldes *feld* mit der festgelegten Größe m. Jedes Element $feld[j]$, $j = 0, \ldots, m-1$, besteht aus folgenden Komponenten:

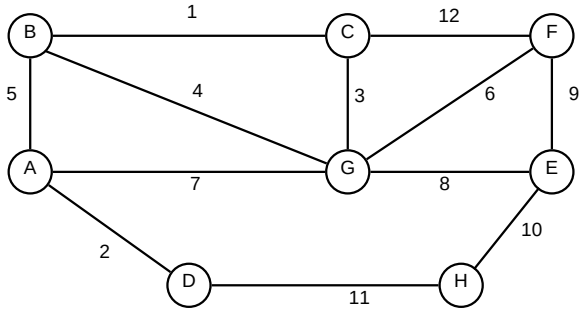
- $feld[j].schluessel$ enthält den Schlüssel;
- $feld[j].zustand$ enthält einen der folgenden Werte:
  - *besetzt*: $feld[j]$ enthält einen Schlüssel;
  - *frei*: $feld[j]$ ist frei und war nie besetzt;
  - *wiederfrei*: $feld[j]$ war schon besetzt, ist aber wieder frei.

Schreiben Sie eine Prozedur $insert(key)$ in Pseudocode, die den Schlüssel key an erstmöglicher Stelle in eine nichtleere Hashtabelle einfügt. Folgende Punkte sind zu beachten:

- Zur Behandlung von Kollisionen ist Double Hashing ohne der Verbesserung nach Brent mit den Hashfunktionen
  $h_1(k) = k \ mod \ m$ und $h_2(k) = (k \ mod \ (m-1)) + 1$ zu verwenden.
- Ist der einzufügende Schlüssel key bereits in der Hashtabelle enthalten, so soll dieser nicht erneut eingefügt werden.
- Sie können davon ausgehen, dass die Hashtabelle noch nicht vollständig befüllt ist und ein freier Platz für den Schlüssel key gefunden werden kann.
- Auf Die Tabellengröße m und das Feld *feld* kann global zugegriffen werden.

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

Gegeben ist der folgende gewichtete ungerichtete Graph G:



a) (8 Punkte) Auf dem gegebenen Graphen G wird die aus dem Skriptum bekannte **Tiefensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge entstehen, wenn die Nachbarn eines Knotens in lexikographischer bzw. in beliebiger Reihenfolge abgearbeitet werden? Kreuzen Sie Zutreffendes an. *Hinweis:* Jede Zeile wird nur dann gewertet wenn sie vollständig richtig ist.

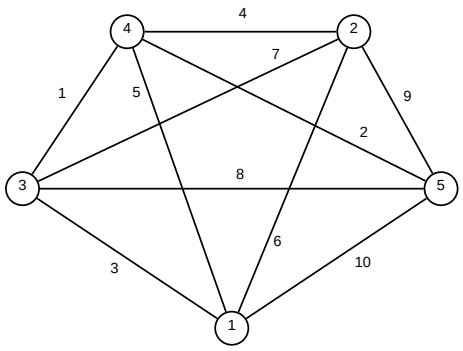
Reihenfolge	lexikograph.	beliebig	keines
A B C D E F G H			
A D H E G B C F			
A G E F C B H D			
A B C F E G H D			

b) (8 Punkte) 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 die Kanten in der Reihenfolge, wie sie zu dem Spannbaum hinzugefügt werden, in die unten angeführte Tabelle ein. Falls Sie einen Startknoten benötigen, wählen Sie D. *Hinweis:* Es werden nicht unbedingt alle Zeilen benötigt. Als Kantenbezeichnung können Sie auch die Gewichte der Kanten eintragen.

Reihenfolge	Kruskal	Prim
1. Kante		
2. Kante		
3. Kante		
4. Kante		
5. Kante		
6. Kante		
7. Kante		
8. Kante		
9. Kante		
10. Kante		

Aufgabe 3.A: Optimierung **(16 Punkte)**

Gegeben ist ein vollständiger, ungerichteter Graph $G = (V, E)$:



Jeder Kante $(u, v) \in E$ ist ein Gewicht $weight_{(u,v)}$ zugewiesen. Gegeben ist ferner folgender Algorithmus:

Was-bin-ich?

```
1: Input: Graph G = (V, E)
2: Output: Kantenmenge F und
3: Output: Feld X[1...|V|]
4: F = {};
5: X[1...|V|] = (0, 0, ..., 0);
6: A(Node4, Node4, 1);
```

Prozedur
A$(v, startnode, i)$

```
1: X[v] = i;
2: i = i + 1;
3: weight_best = ∞;
4: nextnode = NULL;
5: für Knoten w ∈ N(v) {
6:    falls X[w] == 0 ∧
      weight_(v,w) < weight_best  dann
      {
7:       nextnode = w;
8:       weight_best = weight_(v,w);
9:    }
10: }
11: falls nextnode == NULL dann
    {
12:    F = F ∪ (v, startnode);
13: } sonst {
14:    F = F ∪ (v, nextnode);
15:    A(nextnode, startnode, i);
16: }
17: return;
```

- Welche Kanten liefert der Algorithmus **Was-bin-ich?** für den oben angeführten Graphen G in F zurück? Markieren Sie diese Kanten im Graphen G. Geben Sie weiters das Feld X nach der Ausführung des Algorithmus an.
- Beschreiben Sie in einer Phrase, was dieser Algorithmus allgemein für einen beliebigen vollständigen, ungerichteten Graphen macht und in F bzw. X zurückliefert.
- Geben Sie für einen beliebigen vollständigen, ungerichteten Graphen die genaue Anzahl der in F gespeicherten Kanten, in Abhängigkeit der Anzahl der Knoten $|V|$, an.
- Auf welchem aus der Vorlesung bekannten Verfahren beruht der Algorithmus **Was-bin-ich?**

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 2011

10. 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.

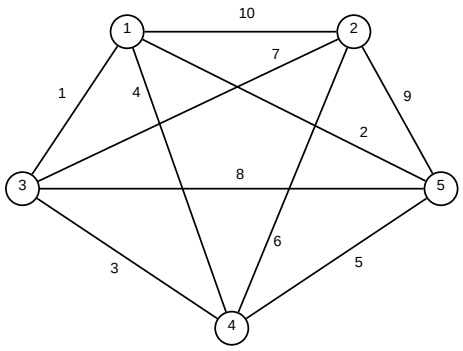
Die Arbeitszeit beträgt 55 Minuten.

	B1:	B2:	B3:	Summe:
Erreichbare Punkte:	16	16	18	50
Erreichte Punkte:				

Viel Glück!

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

Gegeben ist ein vollständiger, ungerichteter Graph $G = (V, E)$:



Jeder Kante $(u, v) \in E$ ist ein Gewicht $weight_{(u,v)}$ zugewiesen. Gegeben ist ferner folgender Algorithmus:

Was-bin-ich?

1: Input: Graph $G = (V, E)$
2: Output: Kantenmenge F und
3: Output: Feld $X[1 \ldots |V|]$
4: $F = \{\}$;
5: $X[1 \ldots |V|] = (0, 0, \ldots, 0)$;
6: A(*Node3*, *Node3*, 1);

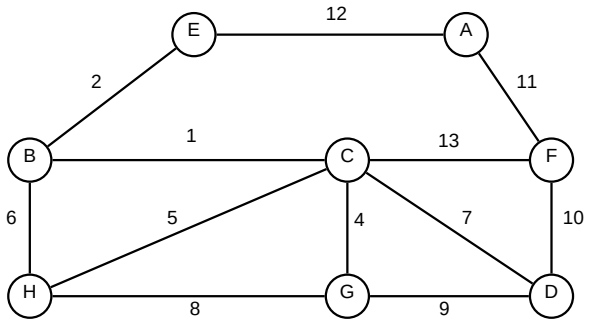
Prozedur
A$(v, startnode, i)$

1: $X[v] = i$;
2: $i = i + 1$;
3: $weight_{best} = \infty$;
4: $nextnode = NULL$;
5: **für** Knoten $w \in N(v)$ **{**
6: **falls** $X[w] == 0 \wedge$ $weight_{(v,w)} < weight_{best}$ **dann** **{**
7: $nextnode = w$;
8: $weight_{best} = weight_{(v,w)}$;
9: **}**
10: **}**
11: **falls** $nextnode == NULL$ **dann** **{**
12: $F = F \cup (v, startnode)$;
13: **} sonst {**
14: $F = F \cup (v, nextnode)$;
15: $A(nextnode, startnode, i)$;
16: **}**
17: *return*;

- Welche Kanten liefert der Algorithmus **Was-bin-ich?** für den oben angeführten Graphen G in F zurück? Markieren Sie diese Kanten im Graphen G. Geben Sie weiters das Feld X nach der Ausführung des Algorithmus an.
- Beschreiben Sie in einer Phrase, was dieser Algorithmus allgemein für einen beliebigen vollständigen, ungerichteten Graphen macht und in F bzw. X zurückliefert.
- Geben Sie für einen beliebigen vollständigen, ungerichteten Graphen die genaue Anzahl der in F gespeicherten Kanten, in Abhängigkeit der Anzahl der Knoten $|V|$, an.
- Auf welchem aus der Vorlesung bekannten Verfahren beruht der Algorithmus **Was-bin-ich?**

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

Gegeben ist der folgende gewichtete ungerichtete Graph G:



a) (8 Punkte) Auf dem gegebenen Graphen G wird die aus dem Skriptum bekannte **Tiefensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge entstehen, wenn die Nachbarn eines Knotens in lexikographischer bzw. in beliebiger Reihenfolge abgearbeitet werden? Kreuzen Sie Zutreffendes an. *Hinweis:* Jede Zeile wird nur dann gewertet wenn sie vollständig richtig ist.

Reihenfolge	lexikograph.	beliebig	keines
A E B C D F G H			
A F D G C H B E			
A F C B H G D E			
A F C D G B E H			

b) (8 Punkte) 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 die Kanten in der Reihenfolge, wie sie zu dem Spannbaum hinzugefügt werden, in die unten angeführte Tabelle ein. Falls Sie einen Startknoten benötigen, wählen Sie A. *Hinweis:* Es werden nicht unbedingt alle Zeilen benötigt. Als Kantenbezeichnung können Sie auch die Gewichte der Kanten eintragen.

Reihenfolge	Prim	Kruskal
1. Kante		
2. Kante		
3. Kante		
4. Kante		
5. Kante		
6. Kante		
7. Kante		
8. Kante		
9. Kante		
10. Kante		

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

a) (12 Punkte) Gegeben ist eine Hashtabelle in Form eines Feldes *feld* mit der festgelegten Größe m. Jedes Element $feld[j]$, $j = 0, \ldots, m-1$, besteht aus folgenden Komponenten:

- $feld[j].schluessel$ enthält den Schlüssel;
- $feld[j].zustand$ enthält einen der folgenden Werte:
  - *besetzt*: $feld[j]$ enthält einen Schlüssel;
  - *frei*: $feld[j]$ ist frei und war nie besetzt;
  - *wiederfrei*: $feld[j]$ war schon besetzt, ist aber wieder frei.

Schreiben Sie eine Prozedur $insert(key)$ in Pseudocode, die den Schlüssel key an erstmöglicher Stelle in eine nichtleere Hashtabelle einfügt. Folgende Punkte sind zu beachten:

- Zur Behandlung von Kollisionen ist Double Hashing ohne der Verbesserung nach Brent mit den Hashfunktionen
  $h_1(k) = k \; mod \; m$ und $h_2(k) = (k \; mod \; (m-1)) + 1$ zu verwenden.
- Ist der einzufügende Schlüssel key bereits in der Hashtabelle enthalten, so soll dieser nicht erneut eingefügt werden.
- Sie können davon ausgehen, dass die Hashtabelle noch nicht vollständig befüllt ist und ein freier Platz für den Schlüssel key gefunden werden kann.
- Auf Die Tabellengröße m und das Feld *feld* kann global zugegriffen werden.

b) (6 Punkte)

Gegeben seien zwei 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 6 **ohne** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
		2				13

- Fügen Sie den Wert 7 **mit** der Verbesserung nach Brent ein.

0	1	2	3	4	5	6
21			3	4		6