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

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

Nachtragstest SS 2014

22. Oktober 2014

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: Vorname:

Matrikelnummer: Unterschrift:

Anzahl abgegebener Zusatzblätter:

Legen Sie während der Prüfung Ihren Ausweis für Studierende vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie bitte dokumentenechte Schreibgeräte (keine Bleistifte!).

Die Verwendung von Taschenrechnern, Mobiltelefonen, Tablets, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

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

Viel Erfolg!

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

Fügen Sie die folgenden Zahlen in die jeweiligen Hashtabellen ein, indem Sie die angegebenen Hashfunktionen und Strategien für die Kollisionsbehandlung benutzen.
Jede Aufgabe wird als richtig gewertet, wenn die Tabelle vollständig und korrekt bearbeitet wurde.

a) (4 Punkte)

Einzufügende Zahl: 14
Kollisionsbehandlung: Lineares Sondieren mit Schrittweite von 1

Hashfunktion:

$$h'(k) = k \bmod 5$$

Hashtabelle:

0	1	2	3	4
5				9

b) (4 Punkte)

Einzufügende Zahl: 21
Kollisionsbehandlung: Quadratisches Sondieren mit $c_1 = c_2 = \frac{1}{2}$

Hashfunktion:

$$h'(k) = k \bmod 7$$

Hashtabelle:

0	1	2	3	4	5	6
14	8	2	3			

c) (4 Punkte)

Einzufügende Zahl: 15
Kollisionsbehandlung: Double Hashing **ohne** der Verbesserung nach Brent

Hashfunktionen:

$$h_1(k) = k \bmod 7$$
$$h_2(k) = (k \bmod 6) + 1$$

Hashtabelle:

0	1	2	3	4	5	6
	22	2			19	

d) (4 Punkte)

Einzufügende Zahl: 15
Kollisionsbehandlung: Double Hashing **mit** der Verbesserung nach Brent
Wird ein bereits vorhandenes Element verschoben, so muss die neue Position dieses Elementes eindeutig gekennzeichnet werden.

Hashfunktionen:

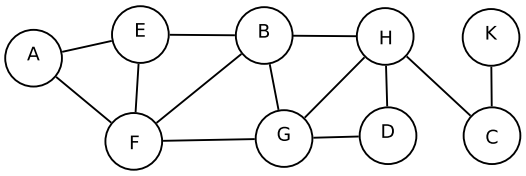
$$h_1(k) = k \bmod 7$$
$$h_2(k) = (k \bmod 6) + 1$$

Hashtabelle:

0	1	2	3	4	5	6
	22	2			19	

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

a) (8 Punkte) Gegeben sei der folgende ungerichtete Graph $G(V, E)$:



Auf diesem Graphen wird eine **Tiefensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge entstehen?

A, E, F, B, G, D, H, C, K ☐ ja ☐ nein
C, H, G, B, F, E, A, D, K ☐ ja ☐ nein
C, H, G, B, F, E, A, K, D ☐ ja ☐ nein
A, E, F, B, G, D, H, C, I ☐ ja ☐ nein

b) (8 Punkte) Vergleichen Sie die Datenstrukturen *einfach verkettete Liste*, *natürlicher binärer Suchbaum*, *sortiertes Feld* und *B-Baum* bezüglich des Aufwandes für

- das Einfügen eines beliebigen Elements und
- die Suche nach dem kleinsten vorhandenen Schlüssel

im Worst-Case in Θ-Notation in Abhängigkeit der Anzahl der gespeicherten Elemente n.

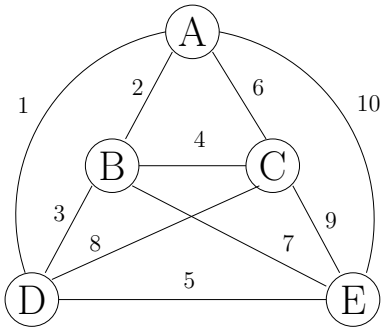
Datenstruktur	Einfügen	Suchen nach Minimum
natürlicher binärer Suchbaum		
AVL-Baum		
verkettete Liste		
sortierte verkettete Liste		

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

Gegeben sei folgender Algorithmus *WasBinIch*, der auf einen vollständigen, ungerichteten, gewichteten Graphen $G(V, E)$ angewendet wird.

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

```
1:  Sortiere die Kanten in E aufsteigend nach Gewicht;
    // Hinweis: Sortierung erfolgt durch Merge-Sort
2:  T = ∅;
3:  d[v] = 0   ∀v ∈ V;
4:  solange |T| < |V| − 1 {
5:      (v, w) = Kante aus E mit dem kleinsten Gewicht;
6:      E = E \ {(v, w)};
7:      falls d[v] < 2 ∧ d[w] < 2 dann {
8:          falls T ∪ {(v, w)} enthält keinen Kreis dann {
9:              T = T ∪ {(v, w)};
10:             d[v] = d[v] + 1;
11:             d[w] = d[w] + 1;
12:         }
13:     }
14: }
15: Finde die Knoten v, w mit v ≠ w ∧ d[v] = d[w] = 1;
16: retourniere T ∪ {(v, w)};
```



Graph G

a) (5 Punkte)

Wenden Sie den obigen Algorithmus *WasBinIch* auf den gegebenen Graphen G an. Geben Sie jeweils die Kanten in der Reihenfolge an, wie sie zu T hinzugefügt werden.

Tragen Sie den Zustand des Feldes d nach der Ausführung des Algorithmus in diese Tabelle ein:

$v \in V$	A	B	C	D	E
$d[v]$					

b) (7 Punkte)

Was berechnet der Algorithmus *WasBinIch*? Kann *WasBinIch* für jeden beliebigen Graphen ein solches Ergebniss berechnen? Falls nicht, geben Sie bitte alle oben genannten Eigenschaften des Graphen an, die Voraussetzung für das korrekte Arbeiten von *WasBinIch* sind?

c) (6 Punkte)

Geben Sie die Gesamtlaufzeit des Algorithmus im Worst-Case in Θ-Notation, in Abhängkeit der Knotenanzahl n an. Begründen Sie ihre Antwort indem Sie kurz beschreiben wodurch sich diese Laufzeit ergibt.

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

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

Nachtragstest SS 2014

22. Oktober 2014

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: Vorname:

Matrikelnummer: Unterschrift:

Anzahl abgegebener Zusatzblätter:

Legen Sie während der Prüfung Ihren Ausweis für Studierende vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie bitte dokumentenechte Schreibgeräte (keine Bleistifte!).

Die Verwendung von Taschenrechnern, Mobiltelefonen, Tablets, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

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

Viel Glück!

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

Fügen Sie die folgenden Zahlen in die jeweiligen Hashtabellen ein, indem Sie die angegebenen Hashfunktionen und Strategien für die Kollisionsbehandlung benutzen.
Jede Aufgabe wird als richtig gewertet, wenn die Tabelle vollständig und korrekt bearbeitet wurde.

a) (4 Punkte)

Einzufügende Zahl: 2
Kollisionsbehandlung: Double Hashing **ohne** der Verbesserung nach Brent

Hashfunktionen:

$h_1(k) = k \bmod 7$
$h_2(k) = (k \bmod 6) + 1$

Hashtabelle:

0	1	2	3	4	5	6
	15	16			12	

b) (4 Punkte)

Einzufügende Zahl: 2
Kollisionsbehandlung: Double Hashing **mit** der Verbesserung nach Brent
Wird ein bereits vorhandenes Element verschoben, so muss die neue Position dieses Elementes eindeutig gekennzeichnet werden.

Hashfunktionen:

$h_1(k) = k \bmod 7$
$h_2(k) = (k \bmod 6) + 1$

Hashtabelle:

0	1	2	3	4	5	6
	15	16			12	

c) (4 Punkte)

Einzufügende Zahl: 9
Kollisionsbehandlung: Lineares Sondieren mit Schrittweite von 1

Hashfunktion:

$h'(k) = k \bmod 5$

Hashtabelle:

0	1	2	3	4
5				14

d) (4 Punkte)

Einzufügende Zahl: 14
Kollisionsbehandlung: Quadratisches Sondieren mit $c_1 = c_2 = \frac{1}{2}$

Hashfunktion:

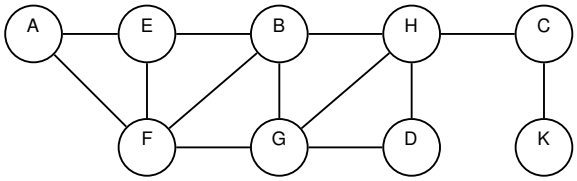
$h'(k) = k \bmod 7$

Hashtabelle:

0	1	2	3	4	5	6
21	8	2	3			

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

a) (8 Punkte) Gegeben sei der folgende ungerichtete Graph $G(V, E)$:



Auf diesem Graphen wird eine **Tiefensuche** durchgeführt. Welche der folgenden Listen von besuchten Knoten können dabei in genau dieser Reihenfolge entstehen?

A, E, F, B, G, D, H, C, K ☐ ja ☐ nein
C, H, G, B, F, E, A, D, K ☐ ja ☐ nein
C, H, G, B, F, E, A, K, D ☐ ja ☐ nein
A, E, F, B, G, D, H, C, I ☐ ja ☐ nein

b) (8 Punkte) Vergleichen Sie die Datenstrukturen *einfach verkettete Liste*, *natürlicher binärer Suchbaum*, *sortiertes Feld* und *B-Baum* bezüglich des Aufwandes für

- das Einfügen eines beliebigen Elements und
- die Suche nach dem kleinsten vorhandenen Schlüssel

im Worst-Case in Θ-Notation in Abhängigkeit der Anzahl der gespeicherten Elemente n.

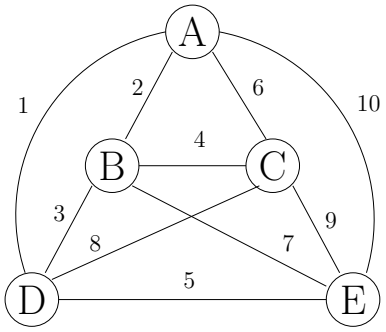
Datenstruktur	Einfügen	Suchen nach Minimum
verkettete Liste		
sortierte verkettete Liste		
natürlicher binärer Suchbaum		
AVL-Baum		

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

Gegeben sei folgender Algorithmus *WasBinIch*, der auf einen vollständigen, ungerichteten, gewichteten Graphen $G(V, E)$ angewendet wird.

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

```
1:  Sortiere die Kanten in E aufsteigend nach Gewicht;
    // Hinweis: Sortierung erfolgt durch Merge-Sort
2:  T = ∅;
3:  d[v] = 0   ∀v ∈ V;
4:  solange |T| < |V| − 1 {
5:      (v, w) = Kante aus E mit dem kleinsten Gewicht;
6:      E = E \ {(v, w)};
7:      falls d[v] < 2 ∧ d[w] < 2 dann {
8:          falls T ∪ {(v, w)} enthält keinen Kreis dann {
9:              T = T ∪ {(v, w)};
10:             d[v] = d[v] + 1;
11:             d[w] = d[w] + 1;
12:         }
13:     }
14: }
15: Finde die Knoten v, w mit v ≠ w ∧ d[v] = d[w] = 1;
16: retourniere T ∪ {(v, w)};
```



Graph G

a) (5 Punkte)

Wenden Sie den obigen Algorithmus *WasBinIch* auf den gegebenen Graphen G an. Geben Sie jeweils die Kanten in der Reihenfolge an, wie sie zu T hinzugefügt werden.

Tragen Sie den Zustand des Feldes d nach der Ausführung des Algorithmus in diese Tabelle ein:

$v \in V$	A	B	C	D	E
$d[v]$					

b) (7 Punkte)

Was berechnet der Algorithmus *WasBinIch*? Kann *WasBinIch* für jeden beliebigen Graphen ein solches Ergebniss berechnen? Falls nicht, geben Sie bitte alle oben genannten Eigenschaften des Graphen an, die Voraussetzung für das korrekte Arbeiten von *WasBinIch* sind?

c) (6 Punkte)

Geben Sie die Gesamtlaufzeit des Algorithmus im Worst-Case in Θ-Notation, in Abhängkeit der Knotenanzahl n an. Begründen Sie ihre Antwort indem Sie kurz beschreiben wodurch sich diese Laufzeit ergibt.