Fakultät IV
Klausur MPGI 2
13.Oktober 2010

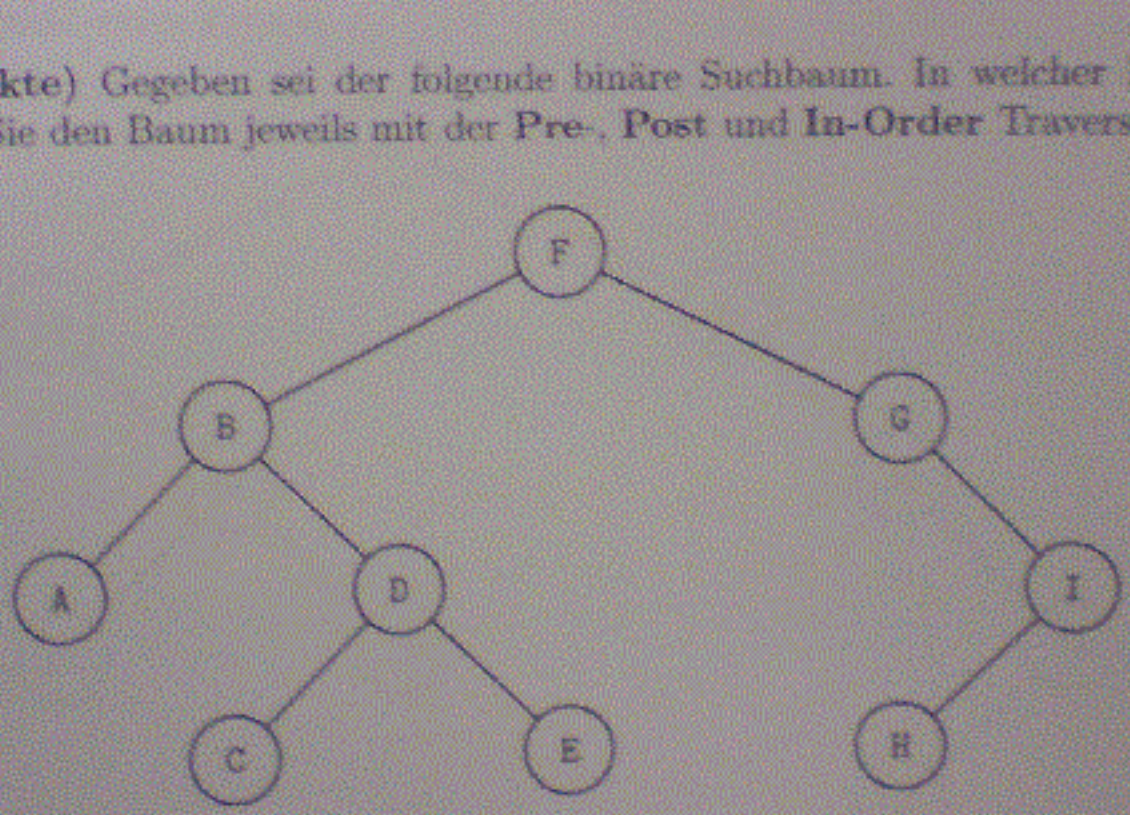
Name: .....

Matr.-Nr. .....

A

1. Aufgabe (9 Punkte): Bäume

1.1. Traversierung (3 Punkte) Gegeben sei der folgende binäre Suchbaum. In welcher Reihenfolge werden Knoten besucht, wenn Sie den Baum jeweils mit der **Pre-**, **Post** und **In-Order** Traversierung durchlaufen?



Pre-Order: F B A D C E G I H ✓

✓

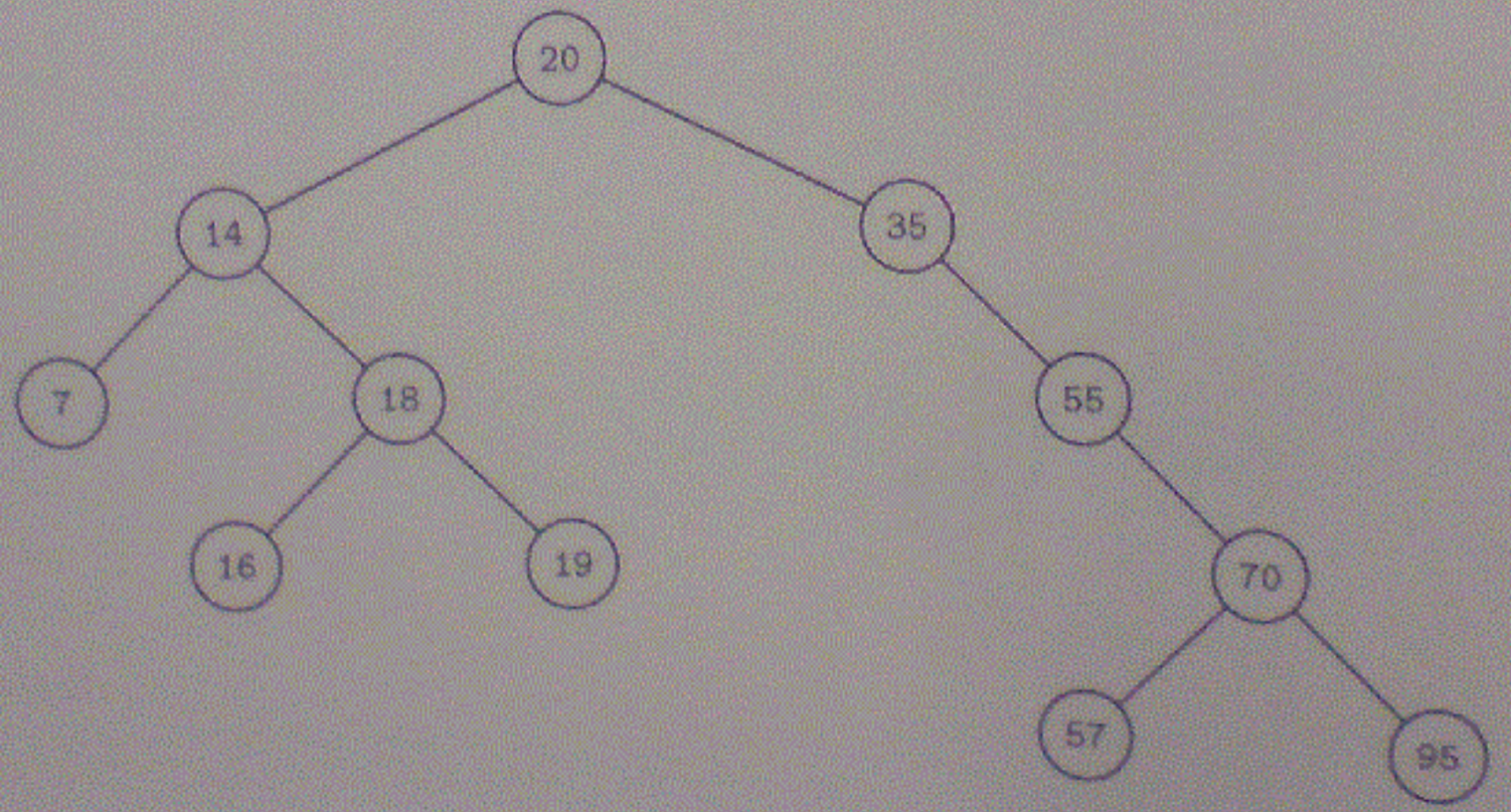
Fakultät IV
Klausur MPGI 2
13 Oktober 2010

Name: .....

Matr.-Nr. .....

A1

1.2. Operationen auf Suchbäumen (6 Punkte) Führen Sie die folgenden Operationen **jeweils** auf dem Suchbaum t aus und zeichnen Sie den jeweils veränderten Baum nach jeder Operation. `insert` fügt einen Knoten in der Baum ein. `delete` löscht einen Knoten.

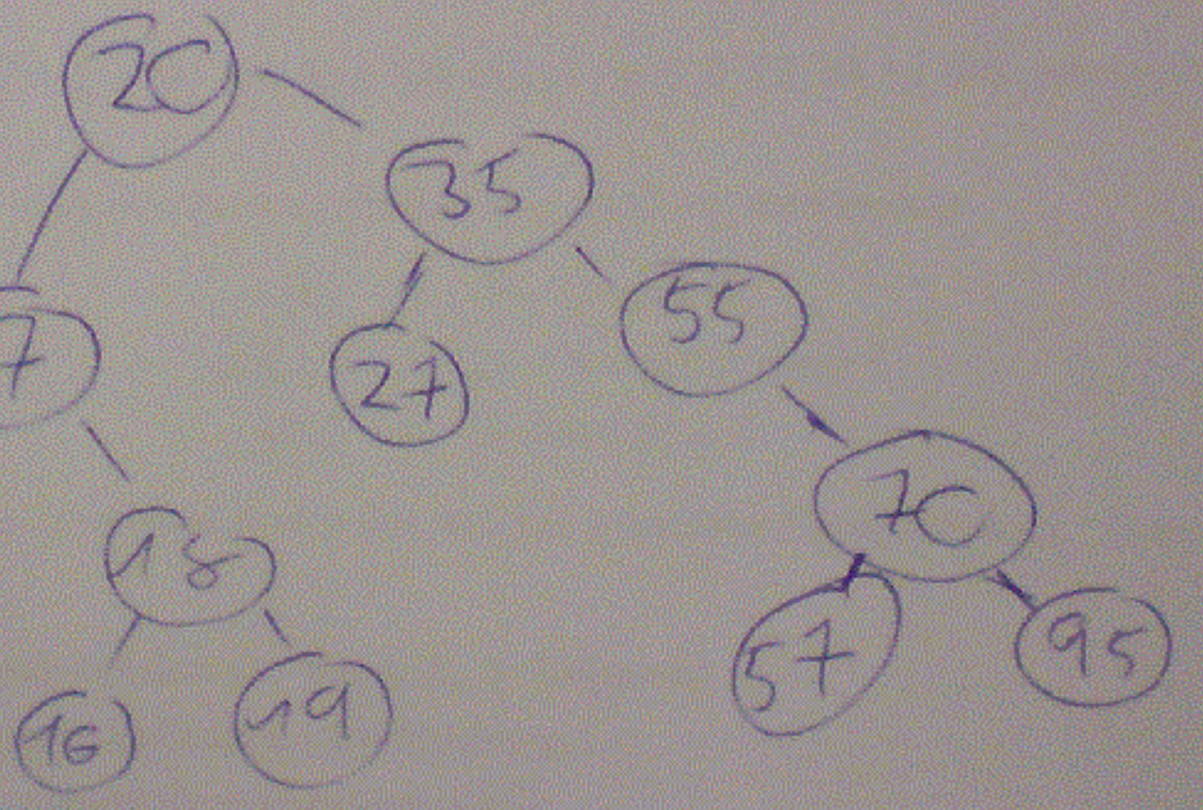


(a) `t.insert(27);`

Name: .....

Matr.-Nr. .....

(b) `t.delete(14);`



(c) `t.insert(42);`

Fakultät IV
Klausur MPGI 2
13.Oktober 2010

Name: .....

Matr.-Nr. .....

A

2. Aufgabe (8 Punkte): Hashing

2.1. Hashtabellen (5 Punkte) Fügen Sie die fünf Schlüssel $14, 8, 0, 12, 34$ in der angegebenen Reihenfolge in e
anfangs leere Hashtabelle ein.

Die folgende Hashfunktion h und Sondierungsfunktion g sind zu verwenden:

$$\begin{aligned} h(k) &= (k+2) \bmod 6 \\ g(k,j) &= (j + 3 * a(k)) \bmod 7 \\ a(k) &= (k \bmod 3) + 1 \end{aligned}$$

Hierbei ist k der einzufügende Schlüssel und j der zuletzt getestete Speicherplatz. Tragen Sie für jeden Schlüs
in die Tabelle ein:

- welche Positionen getestet werden (1. Versuch, 2. Versuch),
- wo und wieviele Kollisionen auftreten (#Kollisionen),
- und wo der Schlüssel letztendlich gespeichert wird.

Schlüssel[k]	1. Versuch	2. Versuch	#Kollisionen
14	4		
8	4	6	1 bei 4

12	2	5	1 Sa 2
34	0		

5/5

- Geben Sie die endgültige Hashtabelle an.

Index	0	1	2	3	4	5	6	7
Schlüssel	34	-	0	-	14	12	8	-

2.2. Weitere Fragen (3 Punkte) Beantworten Sie die folgenden Fragen:

- Wieso ist die in 2.1 verwendete Hashfunktion ungünstig gewählt?

weil 6 keine Primzahl ist und sich Primzahlen besser eignen ✓

- Wie heißt das Sondierungsverfahren aus Aufgabe 2.1?

lineares Sondieren f

- Wie groß ist der Füllfaktor der resultierenden Hashtabelle aus Aufgabe 2.1?

$\frac{n}{N}$ $\frac{5}{10}$ = $\frac{5}{8}$ = 1 Füllfaktor ✓

4/3

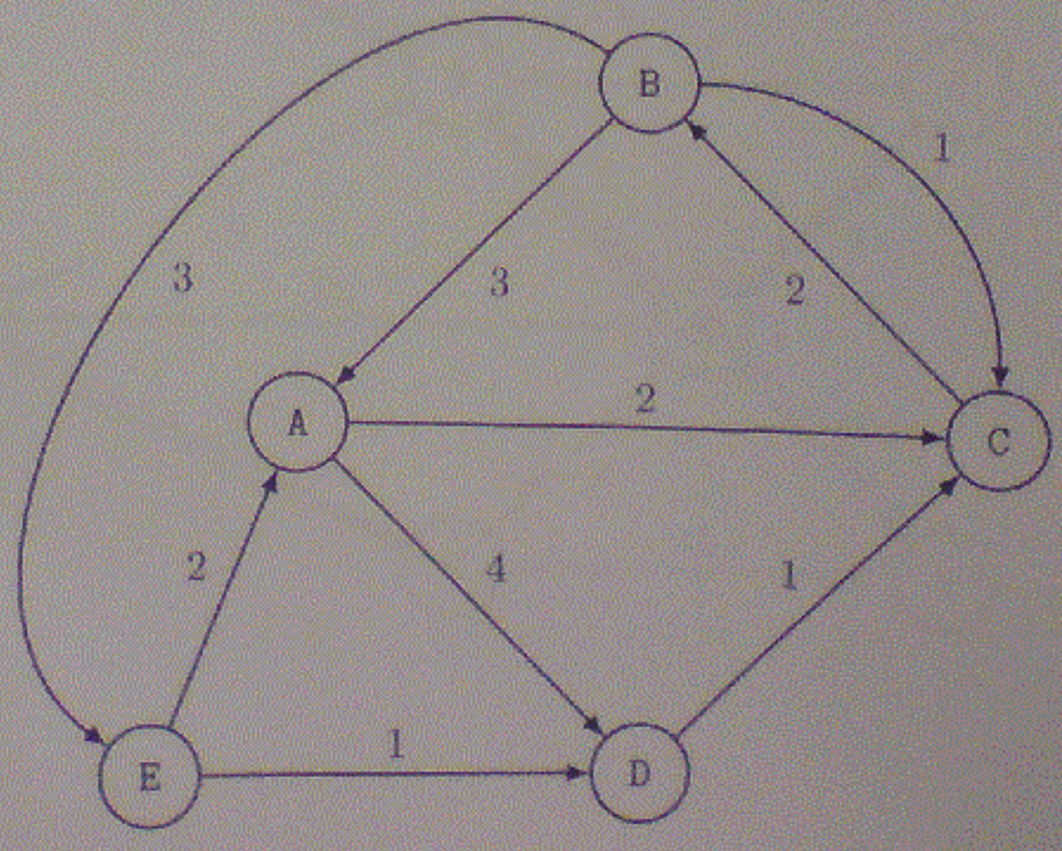
Name: .....

Matr.-Nr. .....

A3

3. Aufgabe (15 Punkte): Graphen

3.1. Darstellung eines Graphen (2 Punkte) Nachfolgend ist ein gerichteter und gewichteter Graph gegeben. Geben Sie die Adjazenzmatrix zu diesem Graphen an.



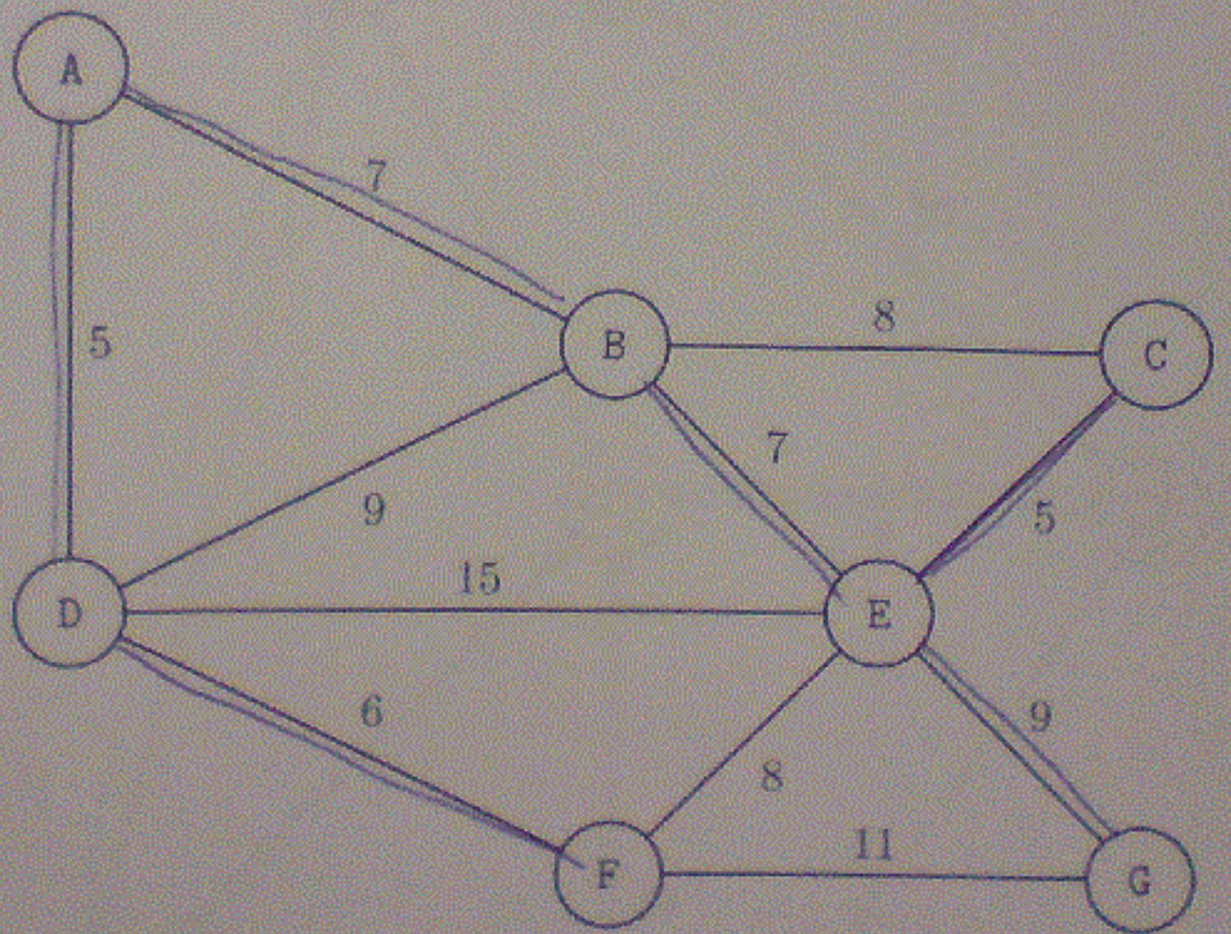
	A	B	C	D	E

Name: .....

Matr.-Nr. .....

A3

3.2. Minimaler Spannbaum (4 Punkte) Gegeben ist der folgende, ungerichtete Graph G. Bitte wenden Sie Kruskals Algorithmus an und schreiben Sie die Reihenfolge der besuchten Kanten auf. Zeichnen Sie zusätzlich den minimalen Spannbaum, der sich ergibt in das Diagramm ein.



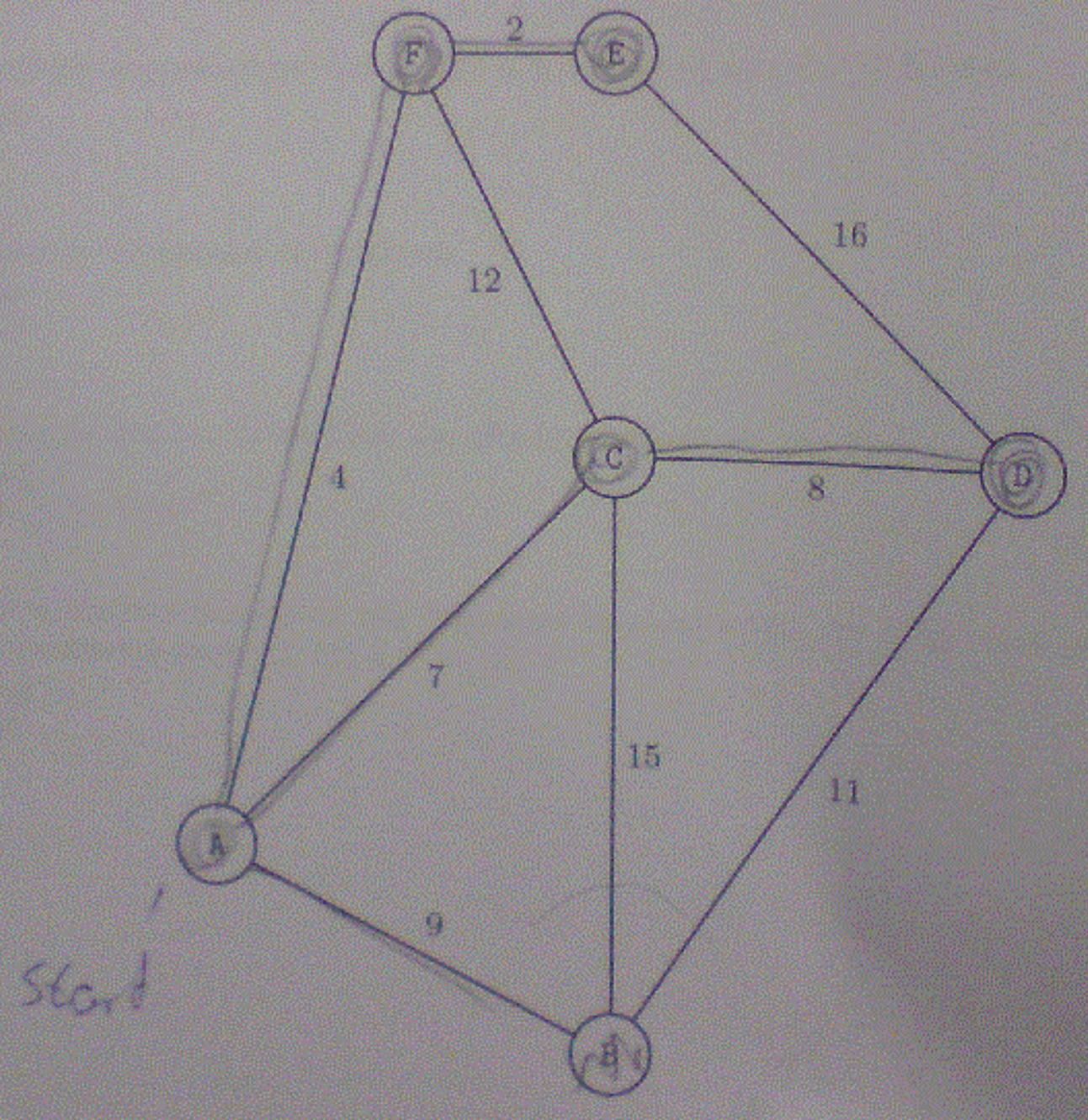
AD (5)

Name: .....

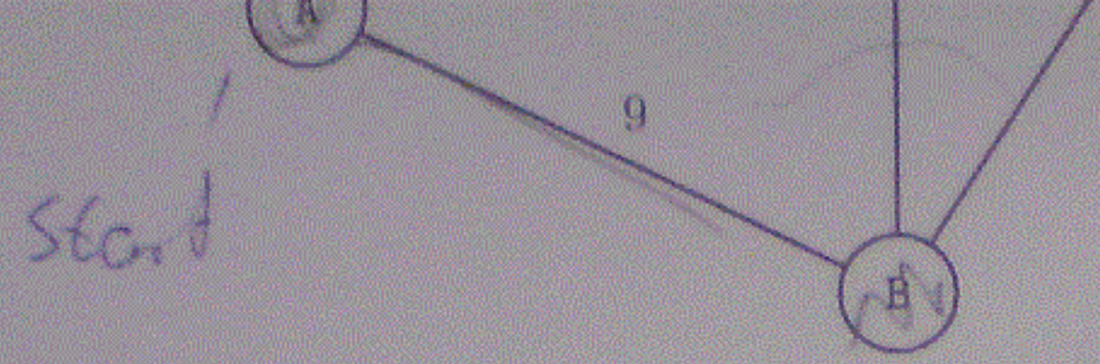
Matr.-Nr. .....

A3

3.3. Dijkstra Algorithmus (5 Punkte) Gegeben ist der folgende, ungerichtete Graph G.



Start



Bestimmen Sie mit Hilfe des Dijkstra-Algorithmus die kürzesten Wege vom Startknoten **A** zu allen anderen Knoten. Verwenden Sie dazu die folgende Tabelle, um dort für jeden Knoten die aktuellen Kosten (d.h. die Länge des jeweils aktuell gefundenen Weges vom Startknoten **A**) anzugeben.

Hinweis: Die erste Zeile (Iteration 0) ist vorgegeben. Bei der 1. Iteration wird also der Knoten **A** betrachtet. Kann sich eine eingetragene Weglänge nicht mehr verbessern, brauchen Sie ab der nächsten Iteration diesen Wert in der entsprechenden Spalte nicht mehr einzutragen.

Iteration	akt. Knoten	Priority-Queue
0	-	A^0
1	A	F C B
2	F	E C B
3	E	C B D
4	C	B D
5	D	B

Iteration	A	B	C	D	E	F
0	0	∞	∞	∞	∞	∞
1	0	9	7	∞	∞	4
2	0	9	7	∞	6	4
3	0	9	7	22	6	4
4	0	9	7	15	6	4
5	0	9	7	15	6	4

Fakultät IV
Klausur MPGI 2
13.Oktober 2010

Name: .....

Matr.-Nr. .....

A3

3.4. Weiterführende Fragen (4 Punkte) Beantworten Sie folgende Fragen, indem Sie die richtige Antwort ankreuzen. Falsch gesetzte Kreuze führen zu Punktabzug. Für nicht gesetzte Kreuze gibt es keinen Abzug. Mindestens gibt es null Punkte für diese Aufgabe.

- Ein vollständiger ungerichteter Graph mit n Knoten hat $n(n-1)/2$ Kanten
  ☒ A. richtig
  ☐ B. falsch

- Die Breitensuche verwendet einen Stack
  ☐ A. richtig
  ☒ B. falsch

- Die Breitensuche hat im Worst Case die gleiche Laufzeit wie die Tiefensuche
  ☒ A. richtig
  ☐ B. falsch

- Es gibt einen binären Suchbaum mit folgenden Knotenwerten, der links- und rechtsvoll (d.h. vollständig) ist. Die Knotenwerte sind: 12, 18, 4, 7, 1, 27, 9
  ☒ A. richtig
  ☐ B. falsch

Fakultät IV
Klausur MPGI 2
13.Oktober 2010

Name: .....

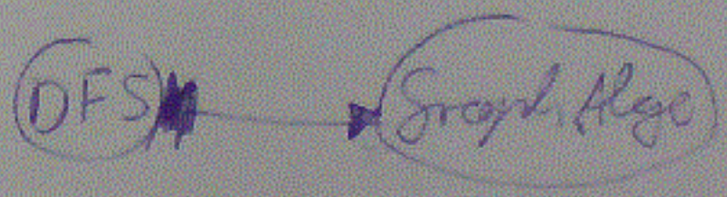
Matr.-Nr. .....

A4

4. Aufgabe (7 Punkte): Topologische Sortierung

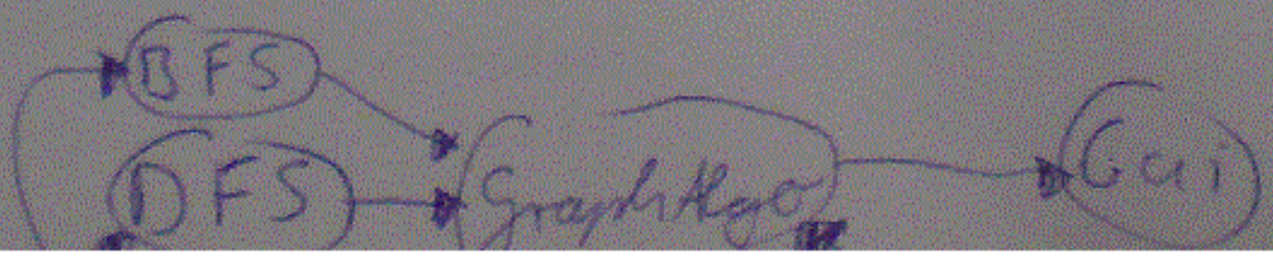
Bei der Übersetzung verschiedener Java-Klassen gibt es folgende Abhängigkeiten:

DFS vor **GraphAlgo**
Node vor **Queue**
Queue vor **Graph**
GraphAlgo vor **GUI**
BFS vor **GraphAlgo**
Graph vor **GUI**
Node vor **Stack**
Queue vor **BFS**
Stack vor **DFS**
Graph vor **GraphAlgo**



Hinweis: Beispielsweise bedeutet die erste Zeile, dass die Klasse **DFS** vor der Klasse **GraphAlgo** übersetzt werden muss.

4.1. Abhängigkeiten als Graphen (3 Punkte) Stellen Sie die Abhängigkeiten, welche beim übersetzen der Java-Klassen vorhanden sind, als Graph dar.



Fakultät IV
Klausur MPGI 2
13.Oktober 2010

Name: .....

Matr.-Nr. .....

4.2. Topologische Sortierung (4 Punkte) Führen Sie auf dem Graph eine topologische Sortierung dur
Zwischenschritte müssen hierbei nicht mit angegeben werden. Geben Sie die dabei entstandene Sortieru

Beginne bei Knoten 100e

6 = GUI
7-Graphalgo
6-BFS
5-Queue
4-DFS
3-Stack

6 = GUI
7 Graph. algo
6 Graph
5 BFS
4 Queue
3 DFS

✓

Name: .....

Matr.-Nr. .....

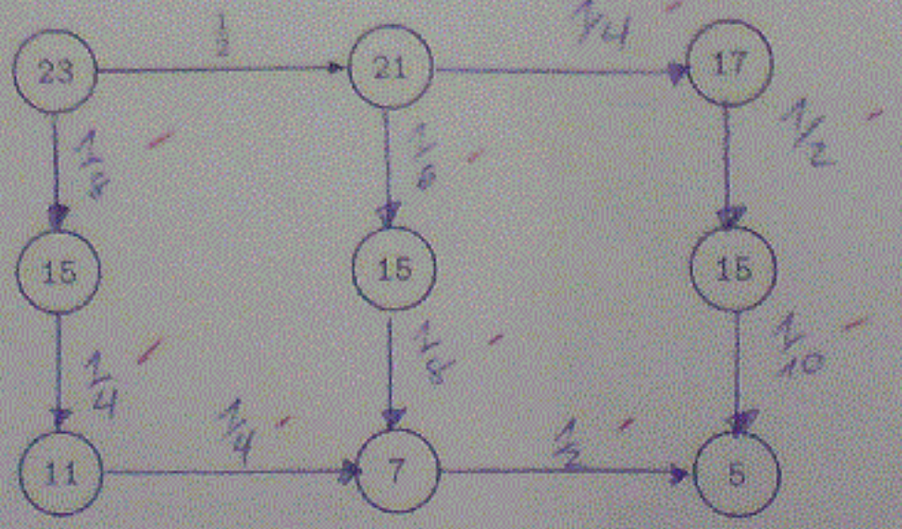
A5

5. Aufgabe (14 Punkte): Flüsse in Netzwerken

Ein Bild soll mit Hilfe des minimalen Schnitts in Vordergrund und Hintergrund geteilt werden. Gegeben ist ein 3x3 Bild mit folgenden Intensitätswerten:

23	21	17
15	15	15
11	7	5

5.1. Bild in Graphen umwandeln (4 Punkte) Formen Sie das Bild in einen Graphen um. Jeder Knoten entspricht einem Pixel. Jeder Knoten kann eine Kante zu seinen vier direkten Nachbarn (oben, unten, links, rechts) haben. Die Richtung der Kante ist von der größeren Intensität zur geringeren. Sind die Intensitäten gleich existiert keine Kante. Die Kapazität ergibt sich aus Eins geteilt durch die Differenz des Startknotens und des Zielknotens. Als Kantengewichte sollen nur diese Kapazitäten eingetragen werden.



5.2. Ford-Fulkerson (8 Punkte) Das Pixel mit der Intensität 23 wurde als Hintergrund (Quelle) und das Pixel mit der Intensität 5 wurde als Vordergrund (Senke) markiert. Führen Sie den Ford-Fulkerson Algorithmus zur Bestimmung des maximalen Flusses aus. Zeichnen Sie dazu in jedem Schritt den Flussgraphen und den Restflussgraphen. Markieren Sie im Restflussgraphen den nächsten zu betrachtenden Pfad. Nutzen Sie dafür die auf den nächsten Seiten vorgegebenen Graphen. Es sind mehr Graphen vorgegeben als für die Lösung der Aufgabe notwendig.

Hinweis: Am Anfang alle Brüche auf einen Nenner zu bringen vereinfacht spätere Rechenoperationen.

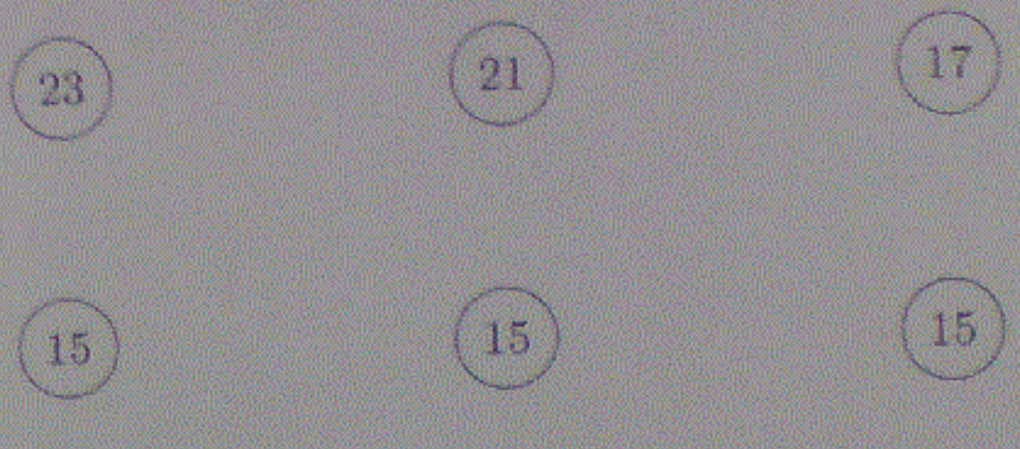
Name: .....

Matr.-Nr. .....

A5

Schritt Flussgraph

Restflussgraph



Name: .....

Matr.-Nr. .....

A5

MinCut einzeichnen, MaxFlow bestimmen (2 Punkte) Zeichnen Sie den minimalen Schnitt in den Graphen von 5.1 ein. Wie groß ist der maximale Fluss zwischen Quelle und Senke?

6. Aufgabe (5 Punkte): Komplexität

Definition der Aufwandsklassen mittels der Landau Symbole $(\mathcal{O}, \Omega, \Theta)$:

$$f \in \mathcal{O}(g) \Leftrightarrow \exists c > 0, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$$

$$f \in \Omega(g) \Leftrightarrow \exists c > 0, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq c \cdot g(n)$$

$$f \in \Theta(g) \Leftrightarrow \exists c_1, c_2 > 0, n_0 \in \mathbb{N} : \forall n \geq n_0 : c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

6.1. Zeitkomplexität (5 Punkte) Beweisen oder widerlegen Sie folgende Aussage mit Hilfe der Definitio
Landau-Symbole.

$$9n^2 + 12n + 8 \in \mathcal{O}(n^3)$$

$\lim_{n\to\infty} \frac{9n^2 + 12n + 8}{n^3} \leq c$? c ?

$9 + \frac{12}{n} + \frac{8}{n^2} \leq cn$

7. Aufgabe (12 Punkte): Java-Programmierung (Priority Queue)

Mit den beiden Klassen `PriorityQueue` und `QueueElem` soll eine Prioritätswarteschlange realisiert werden. Die Prioritäten sollen *aufsteigend sortiert* sein. Die Prioritätswarteschlange soll dabei *einfach verkettet* und *ohne Zyklus* realisiert werden. Die Klasse `QueueElem` ist wie folgt vorgegeben:

```
public class QueueElem {

  public Object value;
  public int priority;
  public QueueElem next;

  public QueueElem(Object value, int priority) {
    this.value = value;
    this.priority = priority;
    this.next = null;
  }

}
```

7.1. Summe berechnen (4 Punkte) Implementieren Sie in der Klasse `PriorityQueue` die Methode `sum`. Diese soll die Summe der Prioritäten berechnen und zurückgeben.

```
public class PriorityQueue {

  private QueueElem head;

  public PriorityQueue(QueueElem head) {
    this.head = head;
```

7.2. Elemente filtern (8 Punkte) Implementieren Sie in der Klasse `PriorityQueue` die Methode `filter`. Diese soll alle Elemente aus der Prioritätswarteschlange entfernen und zurückgeben, die eine Priorität kleiner als ein übergebener Wert haben. Die entfernten Werte sollen wieder als Prioritätswarteschlange zurückgegeben werden. Die Methode erwartet also einen Wert vom Typ `int` als Argument und soll ein Objekt vom Typ `PriorityQueu` als Ergebnis liefern.

```
public class PriorityQueue {

  private QueueElem head;

  public PriorityQueue(QueueElem head) {
    this.head = head;
  }
```