Klausur MPGI 2
8.10.2008

Alexa
Hahne / Rohloff / Tetzlaff / Yilmaz

Name: .....

Vorname: .....

Matr.-Nr.: .....

Bearbeitungszeit: 90 Minuten

- Es ist eine doppelseitig beschriebene DIN-A4 Seite als Hilfsmittel zugelassen, wenn sie Ihren Namen und Ihre Matrikelnummer enthält. Nicht zugelassen sind elektronische Hilfsmittel, wie z. B. Taschenrechner, Handys oder Laptops.
- Benutzen Sie für die Lösung der Aufgaben nur das mit diesem Deckblatt ausgeteilte Papier. **Lösungen, die auf anderem Papier geschrieben werden, können nicht gewertet werden!**
- Schreiben Sie Ihre Lösungen auf das Aufgabenblatt der jeweiligen Aufgabe. Verwenden Sie auch die Rückseiten. **Schreiben Sie keine Lösungen einer Aufgabe auf ein Blatt, das nicht zu dieser Aufgabe gehört!** Wenn Sie zusätzliche, von uns ausgegebene Blätter verwenden, geben Sie unbedingt an, zu welcher Aufgabe die Lösung gehört!
- Schreiben Sie deutlich! Doppelte, unleserliche oder mehrdeutige Lösungen werden nicht gewertet! Streichen Sie gegebenenfalls eine Lösung durch!
- Schreiben Sie nur in **blau** oder **schwarz**. Lösungen, die mit Bleistift geschrieben sind, können nicht gewertet werden!
- Erscheint Ihnen eine Aufgabe mehrdeutig, wenden Sie sich an die Betreuer.
- Sie können die Aufgaben in einer beliebigen Reihenfolge bearbeiten.
- Tragen Sie jetzt (vor Beginn der Bearbeitungszeit) auf *allen* Blättern Ihren Namen und Ihre Matrikelnummer ein.

Aufgabe	Punkte	erreicht
1	12	
2	5	
3	6	
4	14	
5	10	
6	6	
7	7	

1. Aufgabe (12 Punkte): Java-Programmierung (Arrays und Exceptions)

Die Firma Youth verwaltet Jugendherbergen. Eine Jugendherberge besteht immer aus mehreren Häusern, die fortlaufend, mit 0 beginnend, nummeriert sind. Häuser mit gerader Nummer haben zehn Zimmer, die mit ungerader Nummer haben fünf Zimmer. Beim Bau einer Jugendherberge werden in jedes Zimmer sechs Betten gestellt. Aber es können natürlich jederzeit Betten hinzugestellt oder auch Betten aus einem Raum entfernt werden.

Die Firma Youth möchte eine neue Zimmerverwaltungssoftware für ihre Jugendherbergen.

Die Zimmer werden in dem 2-dimensionalen Array `rooms` erfasst (1. Dimension Hausnummer, 2. Dimension Zimmer pro Haus)

Für die Zimmer ist die Klasse `Hostelroom` gegeben.

```
public class Hostelroom {
    public int beds;
    public int freeBeds;

    public Hostelroom(int beds, int freeBeds){
        this.beds = beds;
        this.freeBeds = freeBeds;
    }
}
```

Das Attribut `beds` gibt an, wie viele Betten in diesem Zimmer vorhanden sind. Das Attribut `freeBeds` gibt an, wieviele Betten in diesem Zimmer noch frei sind.

Erweitern Sie in den folgenden Aufgaben die Klasse `Hostel`.

```
public class Hostel {
    public Hostelroom [][] rooms;
    // constructor

    // methods
}
```

1.1. Konstruktor (7 Punkte) Schreiben Sie einen Konstruktor für die Klasse `Hostel`. Achten Sie darauf, dass die Anzahl der Zimmer abhängig von der Hausnummer ist und kein unnötiger Speicherplatz belegt wird. Die Anzahl der Häuser soll als Parameter übergeben werden. Die Zimmer sollen am Anfang alle **nicht** belegt sein und haben alle **sechs** Betten.

1.2. Freie Betten in einem Haus (5 Punkte) Implementieren Sie die Methode `freeBedsInHouse`, welche die Anzahl der freien Betten in einem bestimmten Haus zurückgibt. Die Hausnummer wird als Parameter übergeben. Falls das Haus nicht existiert, soll die Methode eine `HostelException` werfen:

```
class HostelException extends Exception {
    public HostelException(String s){
        super(s);
    }
}
```

2. Aufgabe (5 Punkte): Java-Programmierung (Generics und Interfaces)

Im folgenden soll eine generische Klasse für Mengen implementiert werden. Der Typparameter soll dabei für die Elemente stehen. Diese Elemente sollen das Interface `Comparable<T>` implementieren:

```
public interface Comparable<T> {
      public int compareTo(T obj);
}
```

Die Methode `compareTo` gibt einen negativen `int`-Wert, wenn `obj` kleiner ist als `this`, 0, wenn beide gleich sind und einen positiven `int`-Wert, wenn `obj` größer ist.

2.1. Klassendefinition (2 Punkte) Ergänzen Sie die folgende Klassendefinition für die Klasse `GenericSet`.

```
public class GenericSet _____________________________________________________{
      // assume appropriate code here
}
```

2.2. Klasse für die Elemente schreiben (3 Punkte) Ergänzen Sie die Klasse `MyIntElem`, so dass man `MyIntElem` als Elemente für die Klasse `GenericSet` benutzen kann. Beachten Sie, dass `compareTo` die richtigen Werte zurückliefert. Außerdem soll kein weiteres `return` verwendet werden.

```
public class MyIntElem ____________________________________{
      public int value;
      public int compareTo(_____________________________){
            int res = 0;









            return res;
      }
}
```

3. Aufgabe (6 Punkte): Hashing

3.1. Hashtabellen (5 Punkte) Fügen Sie die drei Schlüssel 18, 8, 5 in der angegebenen Reihenfolge in die gegebene Hashtabelle mit zwei schon eingefügten Elementen ein.

Index	0	1	2	3	4	5	6	7
Schlüssel	-	-	10	-	-	-	6	-

Die folgende Hashfunktion h und Sondierungsfunktion g sind zu verwenden:

$$\begin{aligned} h(k) &= k \bmod 8 \\ g(k,j) &= (j + a(k)) \bmod 8 \\ a(k) &= (5k \bmod 23) + 1 \end{aligned}$$

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

- welche Positionen getestet werden,
- wo und wieviele Kollisionen auftreten,
- und wo der Schlüssel letztendlich gespeichert wird.

Schlüssel[k]	1. Versuch	2. V.	3. V.	#Kollisionen
18				
8				
5				

- Geben Sie die endgültige Hashtabelle an.

3.2. Weitere Fragen (1 Punkt) Beantworten Sie die folgenden Fragen:

- Wie nennt man das in der vorherigen Aufgabe verwendete Sondierungsverfahren?

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

4. Aufgabe (14 Punkte): Graphen

4.1. Darstellung eines Graphen (3 Punkte) Nachfolgend ist ein gerichteter Graph in Form einer Adjazenzliste in reiner Array-Realisierung gegeben. Zeichnen Sie diesen Graphen.

`firstedge`

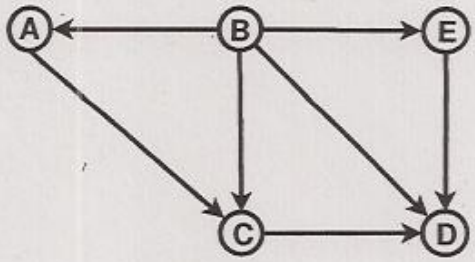
0
2
4
4
6
8

`endnode`

1
2
3
4
4
2
2
0

Zeichnung:

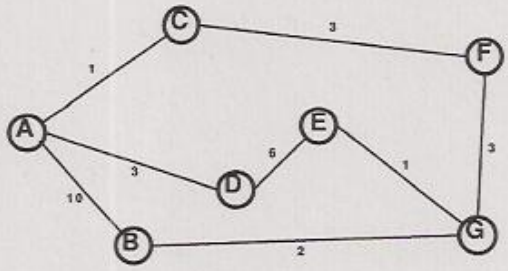
4.2. Topologische Sortierung (1 Punkt) Geben Sie eine topologische Sortierung der Knoten des folgenden Graphen an. Eine gerichtete Kante **A** → **B** bedeutet, dass *B* von *A* abhängig ist.



Sortierung:

4.3. Topologische Sortierbarkeit (1 Punkt) Welche **Bedingungen** muss ein Graph erfüllen, damit er topologisch sortierbar ist?

4.4. Prim Algorithmus (7 Punkte) Gegeben ist der folgende, ungerichtete Graph G.



Bestimmen Sie mit Hilfe des Prim-Algorithmus startend bei Knoten **A** den minimalen Spannbaum. Verwenden Sie dazu die folgende Tabelle, um dort für jeden Knoten die aktuellen Kosten (d.h. das Gewicht der jeweils entsprechenden Kante) anzugeben. Zeichnen Sie den endgültigen minimalen Spannbaum.

Hinweis: Die erste Zeile (Iteration 0) ist vorgegeben. Bei der 1. Iteration wird also der Knoten **A** betrachtet. Nachdem ein Knoten in den Baum eingefügt wurde, brauchen Sie ab der nächsten Iteration diesen Wert in der entsprechenden Spalte nicht mehr einzutragen.

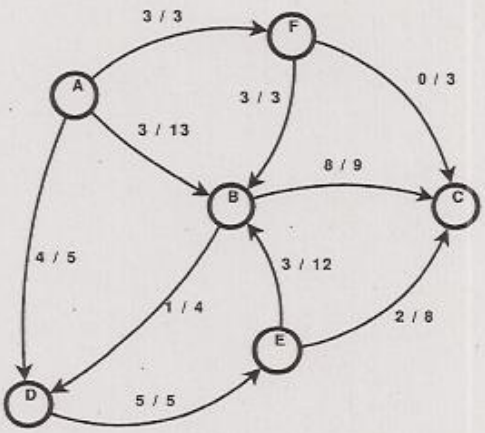
Iteration	akt. Knoten	Priority-Queue
0	-	A^0
1		
2		
3		
4		
5		
6		
7		

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

4.5. Aufwand von Prim (2 Punkte) Schätzen Sie die asymptotische Laufzeit des Algorithmus von Prim im allgemeinen Fall ab. Begründen Sie jeden Schritt ihrer Berechnung.

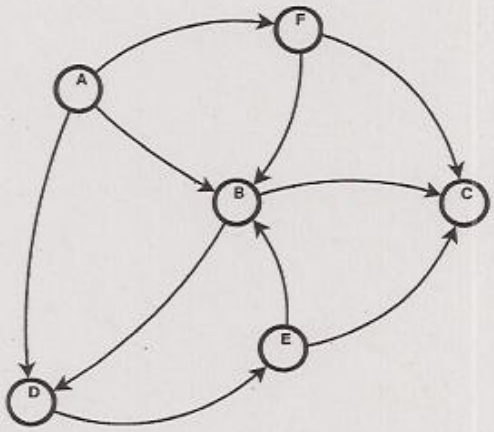
5. Aufgabe (10 Punkte): Flüsse in Netzwerken

5.1. Maximaler Fluss - Minimaler Schnitt (6 Punkte) Betrachten Sie den folgenden gültigen Flussgraphen. Der eingezeichnete Fluss ist nicht maximal.

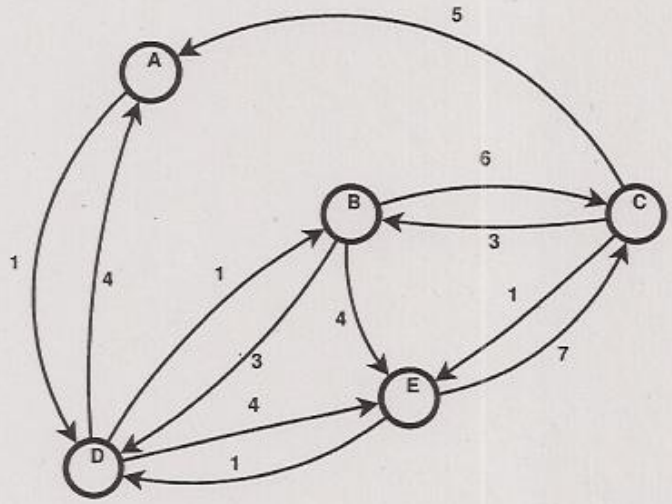


- Ändern Sie den Fluss im Graphen so, dass der Fluss maximal ist,
- zeichnen Sie den Minimalen Schnitt ein,
- und benennen Sie die Quelle und die Senke.

Benutzen Sie bitte die angegebene Zeichnung des Flussgraphen für Ihre Lösung:

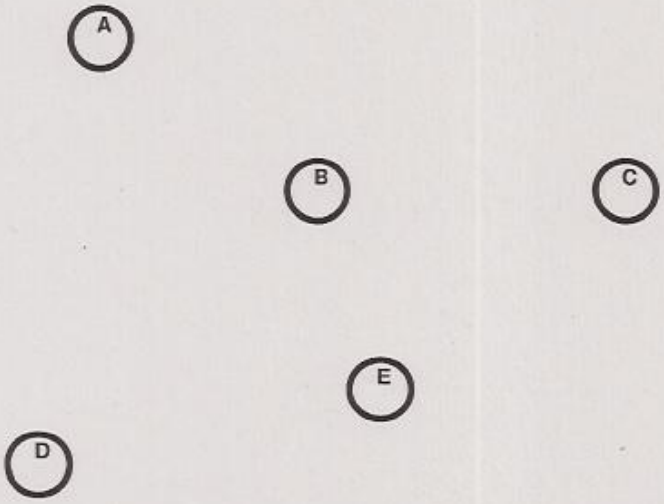


5.2. Restflussgraph (4 Punkte) Betrachten Sie den folgenden Restflussgraphen mit der Quelle A und der Senke C.



- Ist der Fluss des zugehörigen Flussgraphen maximal? Begründen Sie Ihre Antwort.

- Zeichnen Sie einen zugehörigen, gültigen Flussgraphen. Beachten Sie die Einschränkungen, dass die Quelle nur Abflüsse und die Senke nur Zuflüsse enthalten darf, und dass zwischen zwei Knoten immer nur eine Kante existieren darf. Benutzen Sie dazu die Vorlage.



6. Aufgabe (6 Punkte): Numerik

6.1. Implementierung einer Reihenentwicklung (6 Punkte) Die Funktion e^{-x} definieren wir aus numerischen Gründen folgendermaßen :

$$e^{-x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} \ldots + (-1)^n \frac{x^n}{n!} \pm \ldots = \sum_{n=0}^{\infty} (-1)^n \frac{x^n}{n!}$$

Implementieren Sie eine Methode `static double expInv(double x)`, welche die Funktion e^{-x} anhand der oben angegebenen Formel berechnet. Der Algorithmus soll abbrechen, falls das aktuelle Ergebnis sich um weniger als 10^{-5} geändert hat. Hilfsfunktionen und Hilfsmethoden sind für die Implementierung nicht erlaubt.

Hinweis: Falls nötig, kann die Methode `Math.abs()` verwendet werden. Andere Methoden aus der Klasse `Math` sind **nicht erlaubt.**

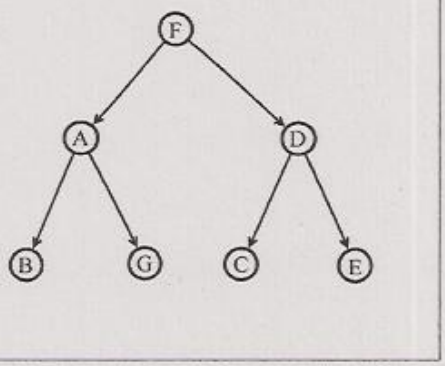
```
static double expInv(double x) {

}
```

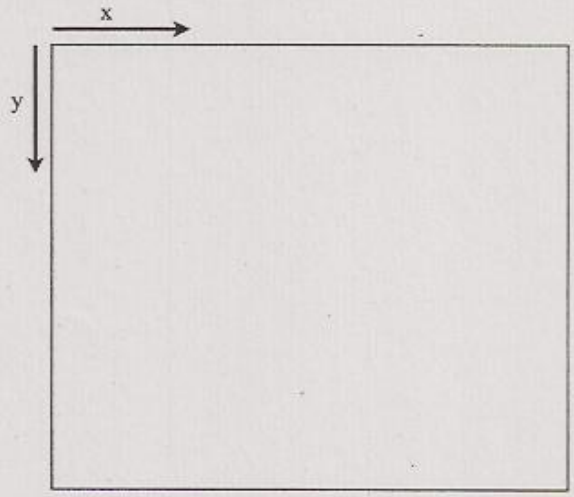
6.2. Zusatzaufgabe Dynamische Programmierung (1 Zusatzpunkt) Nennen Sie eine Stelle in Ihrer Implementierung an der Sie eine Art der dynamischen Programmierung genutzt haben und erklären Sie welchen Vorteil die dynamische Programmierung in diesem Fall hat.

7. Aufgabe (7 Punkte): Bäume

7.1. kD-Bäume (2 Punkte) Nachfolgend ist ein kD-Baum, der die Sortierung von Punkten in einem zweidimensionalen Raum darstellt. Verwenden Sie zur Aufteilung des Raumes den Median und stellen Sie eine mögliche Verteilung der Punkte und die Aufteilung durch Trennlinien dar. Zeichnen Sie im nebenstehenden Feld anschließend die Punkte und die Trennlinien. In Schicht 0 des Baumes wird nach den x-Koordinaten sortiert. Beachten Sie dabei, dass der Koordinatenursprung in dieser Aufgabe links oben liegt.

Baum:





7.2. Binärbäume mit QuickSort (3 Punkte) Gegeben seien die Namen der Mitarbeiter in einer Firma. Fügen Sie mittels Quicksort-Verfahren die Namen lexikografisch (wie im Wörterbuch) in einen binären Suchbaum ein. Verwenden Sie dabei immer das erste Element einer Datenfolge als Pivotelement und den **Namen als Schlüssel**. Zeigen Sie an jedem Schritt die Teillisten.

Lennart, Albert, Lehmann, Klara, Thomas, Müller, Nelson, Xavier

7.3. Einfügen und Löschen (2 Punkte) Nachfolgend ist ein binärer Suchbaum gegeben. Fügen Sie in diesen Baum einen neuen Knoten mit Schlüssel **9** ein, und löschen Sie danach den Knoten mit Schlüssel **4**. Zeichnen Sie den Baum nach jedem Schritt.