

Klausur
Algorithmen und Datenstrukturen
SS 2025 – 14.08.2025

Hinweise:

- Die Bearbeitungszeit beträgt **90 Minuten!**
- Zum Bestehen der Klausur sind 50 Punkte erforderlich.
- Erlaubte Hilfsmittel: keine
- Alle vorgegebenen und zu erstellenden Programmtexte beziehen sich auf die Programmiersprache Java. **Aus der Java-Klassenbibliothek darf nur die Klasse `Math` verwendet werden!**
- Bevor Sie mit der Bearbeitung der Aufgaben beginnen, **müssen** Sie auf allen Blättern Ihren Namen, Ihre Matrikelnummer und Ihren Studiengang eintragen.
- Der Klausurtext enthält ausreichend Platz zur Lösung der Aufgaben. Sollten Sie trotzdem zusätzliches Papier benötigen, wenden Sie sich an die Klausuraufsicht. Die Nutzung eigenen Papiers ist nicht gestattet.
- Sollte Ihre Lösung nicht unmittelbar unter oder neben der Aufgabenstellung stehen, machen Sie bitte einen entsprechenden Hinweis. Streichen Sie diejenigen Teile der von Ihnen geschriebenen Texte deutlich durch, die nicht in die Bewertung eingehen sollen.
- Bitte schreiben Sie deutlich.

Viel Erfolg!

Aufgabe	Maximalpunkte	Erreichte Punkte
1 (Wissen, Multiple Choice)	15	
2 (Komplexität, Rekursion)	15	
3 (Graphen)	18	
4 (Listen)	20	
5 (Binärbaum, Heapeigenschaft)	20	
6 (Suchen)	12	
Bonuspunkte	10	
Summe	110	

Diese Seite ist absichtlich leer!

Aufgabe 1 (Wissen, Multiple Choice)**15 Punkte**

Bearbeiten Sie die folgenden Multiple-Choice-Aufgaben und beachten Sie dabei die folgenden Bewertungsregeln:

- Für jede richtige Antwort erhalten Sie die genannte Anzahl an Punkten.
- Bei Überschreitung der genannten Anzahl an Antworten wird die Teilaufgabe mit 0 Punkten bewertet. Beispiel: zwei Antworten gefordert, drei Antworten gegeben => 0 Punkte!
- Falsche Antworten führen nicht zu Punktabzug.

a) Eine verkettete Liste mit m Einträgen soll an eine verkettete Liste mit n Einträgen angehängt werden. Geben Sie die Laufzeitkomplexität dieser Operation an, wenn beide verketteten Listen ohne ende-Zeiger implementiert wurden.

(1 Antwort, 3 Punkte)

☐	$O(1)$
☐	$O(n)$
☐	$O(m)$
☐	$O(n + m)$
☐	$O(n * m)$
☐	$O(\log m)$

b) Welche Eigenschaften können Beispiele für AVL-Bäume besitzen? (dies bedeutet nicht, dass die Eigenschaft für alle AVL-Bäume gelten muss, sondern nur, ob es ein Beispiel gibt, wo die geforderte Eigenschaft gilt)

(2 Antworten, 2 Punkte pro richtige Antwort)

☐	Der Wurzelknoten hat Balancefaktor -1
☐	Der AVL-Baum besteht aus genau 3 Knoten und alle haben Balancefaktor 0
☐	Der AVL-Baum besteht aus genau 4 Knoten und alle haben Balancefaktor 0
☐	Der AVL-Baum besteht aus genau 5 Knoten und alle haben Balancefaktor 0
☐	Alle Knoten haben Balancefaktor +1
☐	Alle Knoten haben Balancefaktor +2

c) Welche der folgenden Aussagen über Datenstrukturen sind richtig und welche sind falsch?*(5 Antworten, 1 Punkt pro richtige Antwort)*

richtig	falsch	Aussage
☐	☐	In einem Suchbaum sind die Schlüssel von Kindknoten immer kleiner als die der Elternknoten
☐	☐	Ein korrekt implementiertes <code>Comparable</code> -Interface kann die Gleichheit von Objekten mittels <code>compareTo()</code> prüfen
☐	☐	Eine einfach verkettete Liste kann mehrere <code>nächster</code> -Zeiger mit dem Wert <code>null</code> besitzen
☐	☐	Eine verkettete Liste mit <code>ende</code> -Zeiger, aber deren <code>ende</code> -Zeiger <code>null</code> ist, ist zwangsläufig leer
☐	☐	Ein Binärbaum mit 20 Knoten hat maximal Höhe 3

d) Welche der folgenden Aussagen sind korrekt?*(3 Antworten, 1 Punkt pro richtige Antwort)*

☐	Insertionsort und Selectionsort haben die gleiche Worst-Case Komplexität
☐	Heapsort und Bubblesort haben die gleiche Worst-Case Komplexität
☐	Binäre Suche und Interpolationssuche haben die gleiche Worst-Case Komplexität
☐	Pre-Order Traversierung und In-Order Traversierung eines binären Suchbaums haben die gleiche Worst-Case Komplexität
☐	Bubblesort hat die gleiche Worst-Case Komplexität wie Mergesort
☐	Mergesort und Bottom-Up Mergesort haben die gleiche Worst-Case Komplexität

Aufgabe 2 (Komplexität, Rekursion)**15 Punkte**

Bestimmen Sie für die nachstehenden Prozeduren folgende Informationen:

- die Anzahl der Aufrufe von `tuwas()` für $n=4$.
- die Anzahl der Aufrufe von `tuwas()` als Funktion von n . Sie können für die Formeln davon ausgehen, dass n eine Zweierpotenz ist.
- die asymptotische Zeitkomplexität in O-Notation unter der Annahme, dass die **Methode `tuwas()` eine asymptotische Zeitkomplexität von $O(1)$ besitzt.**

```
static void proz1(int n)
{
    for (int i=n; i>=1, i/=2)
        for (int j=0; j<n; j++)
        {
            tuwas();
        }
}
```

```
static void proz2(int n)
{
    if (n > 0)
    {
        proz2(n-1);
        proz2(n-1);
    }

    tuwas();
}
```

a) Notieren Sie die Lösung in der folgenden Tabelle.

Methode	Anzahl Aufrufe für $n=4$	Anzahl Aufrufe als Funktion von n	Zeitkomplexität in O-Notation
<code>proz1</code>			
<code>proz2</code>			

b) Bestimmen Sie die asymptotische Zeitkomplexität von `proz1` in O-Notation, wobei `tuwas()` durch `tuwas(n)` ersetzt wird und `tuwas(n)` eine asymptotische Zeitkomplexität von $O(n)$ besitzt. Begründen Sie Ihre Antwort.

c) Bestimmen Sie die maximale Rekursionstiefe von `proz2` für $n=4$.

Aufgabe 3 (Graphen)**18 Punkte**

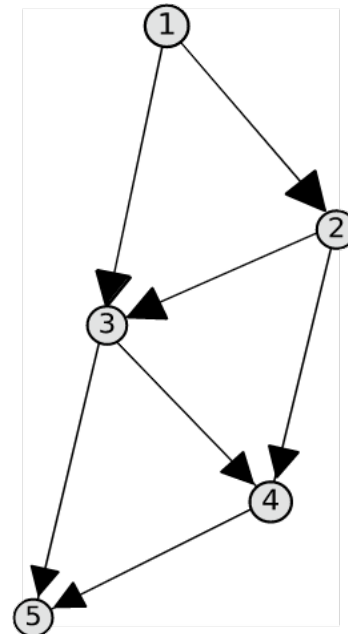
Sei $N > 4$ eine ganze Zahl. Betrachten Sie den gerichteten Graphen, dessen Knoten mit den Zahlen

1 2 3 4 N

bezeichnet werden. Der Graph soll eine gerichtete Kante (d.h. einen Pfeil) von i nach j haben, dann und nur dann wenn

$$j - i = 1 \quad \text{oder} \quad j - i = 2$$

gilt, oder anders gesagt: falls j der erste oder zweite Nachfolger von i ist. Für $N=5$ hat der Graph die ganz rechts abgebildete Form:



a) Geben Sie die Adjazenzmatrix des Graphen für $N=5$ an.

b) Ist dieser Graph für $N = 5$ topologisch sortierbar?

☐ Ja

☐ Nein

c) Ist dieser Graph für jedes N topologisch sortierbar?

☐ Ja

☐ Nein

d) Ist es im Spezialfall $N = 10000$ möglich, eine gerichtete Kante (Pfeil) in den Graphen einzufügen, sodass der resultierende Graph nicht topologisch sortierbar ist? Geben Sie eine konkrete gerichtete Kante (Pfeil) an, wenn dies möglich ist, und erklären Sie Ihre Antwort.

- ☐ Das ist nicht möglich ☐ Das ist möglich, z.B. durch Hinzufügen eines Pfeils von _____ nach _____

Erklärung:

e) In einer konkreten Anwendungssituation kann es sein, dass N ein extrem großer Wert ist. Erklären Sie, ob es dann speicherplatzsparender ist, den Graphen als Adjazenzmatrix oder mit Adjazenzenlisten im Speicher zu halten. Weniger Speicher benötigt die...

- ☐ Adjazenzmatrix ☐ Adjazenzenliste

Begründen Sie Ihre Antwort knapp in einem Satz:

f) Betrachten Sie nun den Graphen im Spezialfall $N = 120$ und entfernen Sie die Knoten mit den Zahlen 59, 60, 61 mitsamt aller in sie eingehenden oder ausgehenden Kanten. Wie viele Zusammenhangskomponenten besitzt der resultierende Graph?

Antwort: _____

Begründen Sie Ihre Antwort knapp in einem Satz:

Aufgabe 4 (Listen)**20 Punkte**

Betrachten Sie die folgende teilweise Implementierung einer einfach verketteten Liste:

```
public class Link<T>
{
    public T daten;
    public Link<T> naechster;

    public Link(T daten, Link<T> naechster){
        this.daten = daten;
        this.naechster = naechster;
    }
}
```

```
public class Liste<T>
{
    public Link<T> anfang;

    public Liste()
    {
        anfang = null; // leere Liste
    }

    private void entfernen(Link<T> zeiger)
    {
        // Aufgabenteil a)
    }

    public void halbieren()
    {
        // Aufgabenteil b)
    }
}
```

- a) Vervollständigen Sie die Methode `entfernen`, die das Nachfolgeelement von `zeiger` aus der Liste entfernt, wenn dieses existiert.

```
private void entfernen(Link<T> zeiger)
{
```

```
}
```

- b) Vervollständigen Sie die Methode `halbieren`, welche jedes zweite Element aus der Liste entfernt.

Hinweis: Nutzen Sie die Methode aus Aufgabenteil a)

```
public void halbieren()  
{
```

```
}
```

Aufgabe 5 (Binärbaum, Heapeigenschaft)**20 Punkte**

Betrachten Sie die folgende teilweise Implementierung eines Heap als Binärbaum:

```
public class HKnoten
{
    public int schluessel;
    public HKnoten linkesKind;
    public HKnoten rechtesKind;
}

public class Heap
{
    private HKnoten wurzel;

    // Methoden zum Einfügen und Entfernen
    // ...

    public int groessterSchluessel() {
        // Aufgabenteil a)
    }

    public boolean istHeap() {
        return istHeap(wurzel);
    }

    private boolean istHeapKnoten(HKnoten knoten) {
        // Aufgabenteil b)
    }

    private boolean istHeap(HKnoten knoten) {
        // Aufgabenteil c)
    }
}
```

- a) Vervollständigen Sie die Methode `groessterSchluessel`. Diese soll den größten Schlüssel im Baum zurückgeben.

Hinweise:

- Sie können davon ausgehen, dass die Wurzel existiert.
- Sie können davon ausgehen, dass der Binärbaum die Schlüsseleigenschaft für einen Heap erfüllt.

```
public int groessterSchluessel()
{
    assert(wurzel != null);
    assert(istHeap());

}
```

- b) Vervollständigen Sie die Methode `istHeapKnoten`, die prüft, ob `knoten` und seine Kinder die Schlüsseleigenschaft eines Heap erfüllen.

```
private boolean istHeapKnoten(HKnoten knoten)
{
```

```
}
```

- c) Vervollständigen Sie die Methode `istHeap`, die rekursiv überprüft, ob alle Knoten im Teilbaum mit der Wurzel `knoten` die Schlüsseleigenschaft eines Heap erfüllen.

Hinweise:

- **Vollständige Ausgeglichenheit und Linksbündigkeit der Blätter müssen nicht geprüft werden!**
- Nutzen Sie die Methode `istHeapKnoten` aus b).

```
private boolean istHeap(HKnoten knoten)
{
```

```
}
```

Aufgabe 6 (Suchen)**12 Punkte**

Betrachten Sie das Array a , gegeben durch

0	1	2	3	4	5	6	7
1	2	3	4	20	21	22	23

Dabei gibt die obere Zeile den Array-Index i an und die untere den Wert (bzw. Schlüssel) $a[i]$, also z.B. $a[0]=1$ und $a[7]=23$. In dieser Aufgabe dürfen Sie voraussetzen, dass eine Schlüsselsuche nur für solche Schlüssel durchgeführt wird, die auch tatsächlich im Array vorliegen.

1. Wenn Sie in dem Array den Algorithmus der **binären Suche** verwenden,...
 1. ...was ist die maximal notwendige Anzahl an Schritten, bis ein Schlüssel garantiert gefunden wurde?
Antwort: _____
 2. ...welche sind die Schlüssel, die auch wirklich diese maximale Anzahl an Schritten erfordern, bis sie gefunden sind?
Antwort: _____
 3. In der Vorlesung wurde die Worst-Case Komplexität der Binärsuche als $O(\log_2 n)$ beschrieben. Welchen Wert hätte n in dem vorliegenden konkreten Anwendungsfall?
Antwort: _____
2. Wenn Sie alternativ in dem Array die **lineare/sequenzielle Suche** verwenden,...
 1. ...was ist die maximal notwendige Anzahl an Schritten, bis ein Schlüssel garantiert gefunden wurde?
Antwort: _____
 2. ...welche sind die Schlüssel, die auch wirklich diese maximale Anzahl an Schritten erfordern, bis sie gefunden sind?
Antwort: _____
3. Das Array ist ja bereits sortiert. Wie viele Vergleiche würde `insertionsort` durchführen, wenn Sie es (dennoch) auf a anwenden?
Antwort: _____

Zusatzblatt