

Prof. Dr. Sándor Fekete
Ramin Kosfeld
Chek-Manh Loi

Klausur
Algorithmen und Datenstrukturen
13.08.2025

Nachname:

Vorname:

Matr.-Nr.:

Studiengang:

☐ Bachelor ☐ Master ☐ Andere:

Hinweise:

- Bitte das Deckblatt in Druckschrift vollständig ausfüllen.
- Die Klausur besteht aus 12 Blättern, bitte auf Vollständigkeit überprüfen.
- Die Bearbeitungszeit für die Klausur beträgt 120 Minuten.
- Erlaubte Hilfsmittel: keine
- Eigenes Papier ist nicht erlaubt.
- Die Rückseiten der Blätter dürfen beschrieben werden.
- Die Klausur ist mit 50 % der Punkte bestanden.
- Antworten, die *nicht* gewertet werden sollen, bitte deutlich durchstreichen. Kein Tippex verwenden!
- Mit *Bleistift* oder in *Rot* geschriebene Klausurteile können nicht gewertet werden.
- Werden mehrere Antworten gegeben, werten wir die mit der geringsten Punktzahl.
- Sämtliche Algorithmen, Datenstrukturen, Sätze und Begriffe beziehen sich, sofern nicht explizit anders angegeben, auf die in der Vorlesung vorgestellte Variante.
- Sofern nicht anders angegeben, sind alle Graphen als einfache Graphen zu verstehen.

Aufgabe	1	2	3	4	5	6	7	8	Σ
Max	15	16	12	16	8	10	11	12	100
Erreicht									

Aufgabe 1: Graphen

(8+2+2+3 Punkte)

- a) Betrachte den Graphen G aus Abbildung 1. Wende Breiten- und Tiefensuche mit Startknoten v_1 auf G an. Falls zu einem Zeitpunkt mehrere Knoten für den nächsten Schritt in Frage kommen, wähle denjenigen mit dem kleinsten Index. Zeichne die resultierenden Bäume in Abbildung 2.

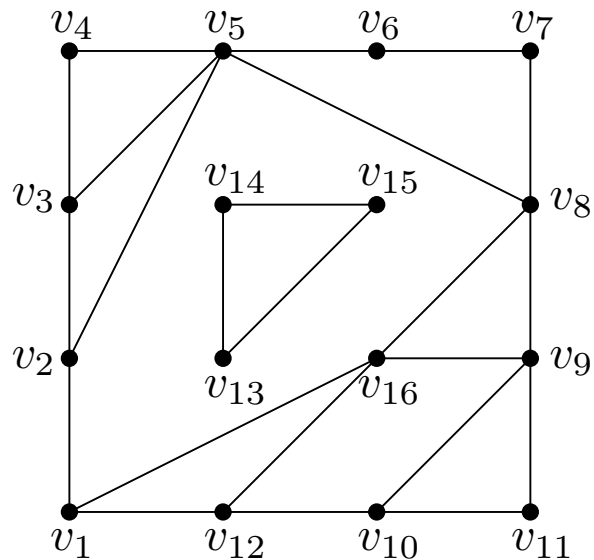


Abbildung 1: Der Graph G .

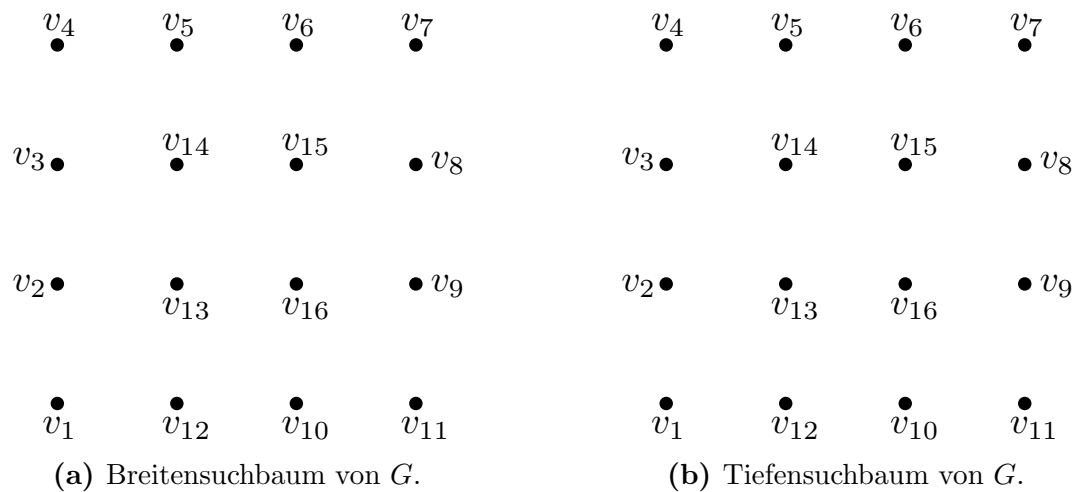


Abbildung 2

- b) Welche Datenstruktur wird für die Tiefensuche benutzt? Welche Laufzeit besitzt Tiefensuche für einen Graphen mit n Knoten und m Kanten?
- c) Welche zwei Eigenschaften müssen in einem Graphen G erfüllt sein, damit in G eine Eulertour (keine Eulerweg) existiert?
- d) Sei I ein Graph mit mindestens 3 Knoten. Zeige oder widerlege: Existiert in I ein Hamiltonkreis, dann gibt es zwischen jedem Paar von Knoten (u, v) einen Hamiltonpfad, der in u startet und in v endet.

Aufgabe 2: Dynamische Datenstrukturen

(7+4+3+2 Punkte)

- a) Füge folgende Elemente der Reihe nach in einen anfangs leeren AVL-Baum ein. Gib den Baum nach jeder Einfüge- und Restructure-Operation an.

1, 2, 3, 4, 5, 6, 7, 8

b) Sei B ein *vollständiger* binärer Baum der Höhe h . Zeige mithilfe von vollständiger Induktion, dass für die Anzahl der Knoten n in B gilt: $n = 2^h - 1$.

c) Zeige, dass das Erstellen eines binären Suchbaums aus einer unsortierten Menge von vergleichbaren Elementen $\Omega(n \log n)$ Zeit benötigt.

d) Beschreibe kurz, wie in verketteten Listen in $\mathcal{O}(1)$ Zeit ein Element eingefügt werden kann.

Aufgabe 3: Wachstum von Funktionen

(4+3+5 Punkte)

a) Seien $f, g, h : \mathbb{N} \rightarrow \mathbb{R}$ Funktionen.

Zeige oder widerlege: $f(n) \in O(g(n))$ und $g(n) \in \Omega(h(n)) \Rightarrow f(n) \in \Omega(h(n))$

b) Bestimme geeignete Konstanten, um zu zeigen, dass folgende Aussage korrekt ist.

$$f(n) := 2n(n-3) + n^3 - 8 \in \mathcal{O}(n^3)$$

c) Ordne folgende Klassen nach Inklusion. Markiere außerdem identische Klassen.

(Hinweis: Schreibe $A = B$, wenn A und B identisch sind. Ansonsten $A \subset B$, wenn jedes Element aus A auch in B vorkommt.)

$$O(15), \quad O\left(\frac{n}{4}\right), \quad O(2^n), \quad O(n^3), \quad O(n \log n), \quad O(n - \log n), \quad O(\log n), \quad O(3.5^n)$$

Aufgabe 4: Rekursionen

(4+2+10 Punkte)

a) Wie lautet das Mastertheorem aus der Vorlesung?

b) Kreuze *wahr* an, wenn das Mastertheorem auf die gegebene Rekursionsgleichung angewendet werden kann (und sonst *falsch*). Eine Begründung ist nicht erforderlich.

(i) $T(n) = 3 \cdot T\left(\frac{n}{3}\right) + 2n + T\left(\frac{9}{n}\right)$

☐ wahr

☐ falsch

(ii) $T(n) = n \cdot T\left(\frac{n}{2}\right) + n$

☐ wahr

☐ falsch

c) Bestimme das asymptotische Wachstum der folgenden Funktionen mithilfe des in der Vorlesung vorgestellten Mastertheorems. Gib beim Anwenden alle im Mastertheorem auftretenden Parameter an.

(i)

$$U(n) = 7n + 64 \cdot U\left(\frac{n}{4}\right) - 19 \log n - 5 + n^2$$

(ii)

$$T(n) = 3 \cdot T\left(\frac{n}{3}\right) + \sqrt{n} + 54 \cdot T\left(\frac{n}{9}\right) + 7n^2$$

(iii)

$$T(n) = T\left(\frac{n}{9}\right) + 4 \cdot T\left(\frac{n}{81}\right) + 8\sqrt{n}$$

Aufgabe 5: Mediane

(4+4 Punkte)

a) Welche Schritte werden benötigt, um das Rang- k Element in einer Menge von n paarweise verschiedenen Zahlen in $O(n)$ Zeit zu finden? Kreuze in jeder Teilaufgabe den richtigen Schritt an.

(i) Teile die n Zahlen gleichmäßig in

- ☐ $t = \lceil n/2 \rceil$ viele Gruppen auf.
- ☐ $t = \lceil n/3 \rceil$ viele Gruppen auf.
- ☐ $t = \lceil n/5 \rceil$ viele Gruppen auf.

(ii) Suche in jeder Gruppe j ($1 \leq j \leq t$)

- ☐ das Minimum m_j .
- ☐ den Median m_j .
- ☐ das Maximum m_j .

(iii) Sei M die Menge aller m_j mit $1 \leq j \leq t$. Suche in M

- ☐ das Rang- $\lceil t/2 \rceil$ Element x .
- ☐ das Rang- $\lceil \sqrt{t} \rceil$ Element x .
- ☐ das Rang- $\lceil \log_2 t \rceil$ Element x .

(iv) Angenommen, es gibt nun $(i - 1)$ Zahlen kleiner als x und es gilt $k > i$. Um das Rang- k Element zu finden, suchen wir in der Menge der Zahlen größer als x rekursiv nach dem

- ☐ Rang- $(k - i)$ Element.
- ☐ Rang- k Element.
- ☐ Rang- $(n - i)$ Element.

b) Sei $\ell \in \{1, \dots, n\}$ und X ein unsortiertes Array von n paarweise verschiedenen Zahlen.

Zeige: Die ℓ kleinsten Zahlen aus X können in Zeit $O(n + \ell \log \ell)$ in sortierter Reihenfolge ausgegeben werden. (Hinweis: Die Laufzeit zum Finden eines Rang- k -Elementes kann ohne Beweis aus der Vorlesung übernommen werden.)

Aufgabe 6: Sortieren

(6+4 Punkte)

- a) Sortiere das Array A aus Abbildung 3 mit dem Algorithmus MERGESORT. Gib (separat und in chronologischer Reihenfolge, also **nicht** für mehrere Teilmengen im selben Schritt) das Array A nach jedem Aufruf von MERGE an. Elemente, die nicht zum im jeweiligen Mergeschritt betrachteten Teilarray gehören, müssen nicht angegeben werden. Die Aufrufe von MERGE auf einem Teilarray der Länge 1 müssen nicht angegeben werden. Nutze zur Bearbeitung der Aufgabe die in Abbildung 3 gegebene Tabelle.

A	7	1	8	6	4	2	5	3
1. A								
2. A								
3. A								
4. A								
5. A								
6. A								
7. A								

Abbildung 3: MERGESORT auf dem Array A .

- b) Fülle die Tabelle in Abbildung 4 aus. (Hinweis: Zellen mit $\times$ müssen nicht ausgefüllt werden.)

	Best-Case	Average-Case	Worst-Case	Stabil?
Mergesort				
Bubblesort	$\times$	$\times$		
Quicksort				

Abbildung 4: Laufzeiten und Stabilität von Sortieralgorithmen.

Aufgabe 7: Algorithmenentwurf

(9+2 Punkte)

- a) Entwirf einen Linearzeitalgorithmus (in Pseudocode!), der als Eingabe eine einfach verkettete Liste L bekommt und eine einfach verkettete Liste L' ausgibt, welche die Elemente aus L in umgekehrter Reihenfolge enthält. So soll das letzte Element in L das erste Element in L' sein usw.

Zeige, dass dein Algorithmus eine lineare Laufzeit besitzt.

- b) Kann es für die Aufgabe aus a) einen korrekten Algorithmus geben, der schneller ist als Linearzeit? Begründe deine Antwort in einem Satz.

Aufgabe 8: Kurzfragen**(2×6 Punkte)**

Kreuze die korrekten Aussagen an. Es gibt nur Punkte für vollständig korrekt angekreuzte Teilaufgaben. (Hinweis: In jeder Teilaufgabe ist immer mindestens eine Aussage korrekt.)

- a) Es gibt AVL-Bäume mit $n > 2$ Knoten der Höhe h , die...
- ... 3^h Knoten besitzen. ☐
 - ... $\Omega(h)$ Restructureoperationen nach einer Einfügeoperation benötigen. ☐
 - ... $\Omega(h)$ Restructureoperationen nach einer Löschoperation benötigen. ☐
- b) Aus einer unsortierten Menge von n vergleichbaren Elementen kann man in $O(n)$ Zeit...
- ... eine einfach verkettete Liste erstellen. ☐
 - ... einen Max-Heap erstellen. ☐
 - ... eine doppelt verkettete Liste erstellen. ☐
- c) Jeder Pfad in einem Graphen G ...
- ... besucht keinen Knoten doppelt. ☐
 - ... besucht jede Kante höchstens einmal. ☐
 - ... ist ein Hamiltonpfad. ☐
- d) Quicksort...
- ... ist ein vergleichsbasierter Sortieralgorithmus. ☐
 - ... nutzt das Prinzip „Teile und Herrsche“ (Divide and Conquer). ☐
 - ... hat eine Worst-Case-Laufzeit von $O(n^2)$. ☐
- e) Wird eine Subroutine, welche die Laufzeit $O(n)$ besitzt, $O(\log n)$ mal wiederholt, so ist die asymptotische Gesamtlaufzeit...
- ... unbekannt, da zunächst die absolute Laufzeit bekannt sein muss. ☐
 - ... $O(n)$. ☐
 - ... $O(n \log n)$. ☐
- f) Die Adjazenzmatrix für einen einfachen Graphen mit n Knoten und m Kanten...
- ... benötigt immer $O(m)$ viel Speicherplatz. ☐
 - ... gibt an, welche Kanten an welchen Knoten anliegen. ☐
 - ... enthält auf der Diagonalen ausschließlich Nullen. ☐

Viel Erfolg ☺