

Nachname (Druckschrift): _____

Vorname (Druckschrift): _____

Matrikelnummer: _____

Studiengang: _____

Die folgenden Hinweise sind zu beachten:

- Schreiben Sie Ihren Namen **nur auf dieses Titelblatt**. Sollten Sie Zusatzpapier bekommen, vermerken Sie darauf unbedingt Ihre Klausurnummer **0001**.
- Merken oder notieren Sie sich Ihre Klausurnummer **0001**, da nur unter dieser Nummer die Ergebnisse veröffentlicht werden.
- Diese Klausur ist für die Prüfungsvariante **Algo-1 (beide Teile)** und besteht aus den Aufgaben **1 – 10**.
- Legen Sie Ihre **Goethe-Card** deutlich sichtbar an Ihren Platz, damit wir während der Klausur Ihre Identität überprüfen können.
- Überprüfen Sie bitte, ob die Klausur aus ? durchnummerierten **Vorder- und Rückseiten** besteht. Beachten Sie, dass die Aufgabenstellungen sowohl auf den Vorder- als auch auf den Rückseiten stehen.
- Es dürfen nur **dokumentenechte Stifte** in den Farben blau und schwarz verwendet werden. Insbesondere ist die Nutzung von Tintenlöschern und Tipp-Ex untersagt. Zugelassene Hilfsmittel:
1 Blatt DIN A4 mit handschriftlichen Notizen (beidseitig).
- Das Mitbringen nicht zugelassener Hilfsmittel stellt eine Täuschung dar und führt zum Nichtbestehen der Klausur. **Schalten Sie bitte deshalb alle elektronischen Geräte, insbesondere Handys und Smartwatches, vor Beginn der Klausur aus.**
- Sollte der Platz unter einer Aufgabe nicht ausreichen, **nutzen Sie bitte die Zusatzblätter am Ende**. Weitere Blätter sind bei Bedarf erhältlich.
- Wenn sich Ihre Lösung zu einer Aufgabe teilweise oder ganz auf Zusatzblättern befindet, vermerken Sie dies entsprechend bei der Aufgabe und auf dem Zusatzblatt.
- Werden zu einer Aufgabe zwei oder mehr Lösungen angegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für **eine** Lösung. Begründungen sind nur dann notwendig, wenn die Aufgabenformulierung dies explizit verlangt.
- Die Klausur ist mit Sicherheit bestanden, wenn mindestens **50%** der Höchstpunktzahl (ohne Bonifikation aus den Übungen) erreicht wird. Die Bearbeitungszeit beträgt **180 Minuten**.

Viel Erfolg! 🍀



**Diese Seite ist nur für den internen Gebrauch bestimmt.
Bitte nicht beschreiben.**



**Diese Seite ist nur für den internen Gebrauch bestimmt.
Bitte nicht beschreiben.**

Aufgabe	1	2	3	4	5	6	7	8	9	10
Erreichbar	22	18	14	16	17	13	22	15	14	9
Erreicht										

Klausur	Bonifikation

- a) Geben Sie jeweils eine Funktion $f : \mathbb{N} \mapsto \mathbb{R}$ an, welche die angegebenen Wachstumseinschränkungen einhält.

$f \in o(n)$ $f \in \omega(\log n)$ $f(n) =$ _____

$f \in \mathcal{O}(n^2)$ $f \in \Omega(n \log n)$ $f(n) =$ _____

$f \in \Theta(f^2)$ $f(n) =$ _____

↑↑↑ _____ / 6 Punkt(e) ↑↑↑

- b) Lösen Sie die folgenden Rekursionsgleichungen auf. Für alle Rekursionsgleichungen gilt $T(1) = 1$. Nehmen Sie vereinfachend an, dass n so gewählt ist, so dass sämtliche Divisionen restfrei aufgehen.

$T(n) = 2 \cdot T(\frac{n}{2}) + n$ $T(n) = \Theta(\text{_____})$

$T(n) = 9 \cdot T(\frac{n}{3}) + 2n + 1$ $T(n) = \Theta(\text{_____})$

$T(n) = T(\frac{3}{6}n) + \sqrt{n}$ $T(n) = \Theta(\text{_____})$

↑↑↑ _____ / 6 Punkt(e) ↑↑↑



- c) Geben Sie für die drei folgenden Algorithmen in Pseudo-Code jeweils die Laufzeit in Θ -Notation an.

```

for  $i = 0 \dots n$  do
   $j = i$ ;
  while  $j > 0$  do
     $j = j - 1$ ;

```

$\Theta(\text{_____})$

```

if  $n > 2022$  then
   $i = n * n$ ;
  while  $i > 2$  do
     $i = \lfloor i/2 \rfloor$ ;

```

$\Theta(\text{_____})$

```

 $i = 1$ ;  $j = n$ ;
while  $i < j$  do
   $i = i + 1$ ;
  while  $2j \geq n$  do
     $j = j - 1$ ;

```

$\Theta(\text{_____})$

↑↑↑ _____ / 6 Punkt(e) ↑↑↑

- d) Sei $A[1 \dots n]$ ein Array mit Zahlen. Die Rekursion `rek` wird mit folgendem Algorithmus in Pseudo-Code berechnet:

```

int rek(int a, int b) {
  if (a == b) return A[a];
  int m =  $\lfloor (a+b)/2 \rfloor$ ;
  int p = rek(a, m);
  int q = rek(m+1, b);
  if (p >= q) return p;
  return q;
}

```

- i) (2 Punkte) Welchen Wert berechnet der Aufruf `rek(1,n)`?

`rek(1,n)` berechnet _____ .

- ii) (2 Punkte) Welche der folgenden Rekursionsgleichungen beschreibt die Laufzeit des Aufrufs `rek(1,n)`? Kreuzen Sie genau eine Antwortmöglichkeit an.

$T(n) = 2 \cdot T(n/2) + O(n)$

$T(n) = T(n/2) + O(n)$

$T(n) = 2 \cdot T(n/2) + O(1)$

$T(n) = T(n-1) + O(1)$

↑↑↑ _____ / 4 Punkt(e) ↑↑↑



- a) Gegeben sei das Array $H[1 \dots n]$ eines *Max-Heaps* mit $n = 10$ paarweise verschiedenen Zahlen.
- i) **(4 Punkte)** Geben Sie die Menge aller Indizes aus $\{1, \dots, 10\}$ an, an denen in H die *weitgrößte* Zahl stehen kann.
- ii) **(5 Punkte)** Geben Sie die Menge aller Indizes aus $\{1, \dots, 10\}$ an, an denen in H die *kleinste* Zahl stehen kann.

↑↑↑ _____ / 9 Punkt(e) ↑↑↑



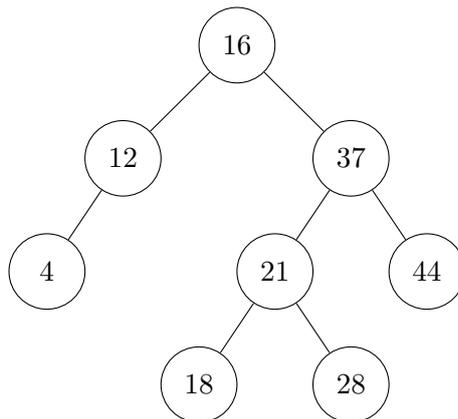
- b) Erweitern Sie die *Max-Heap*-Datenstruktur mit den Operationen `insert` und `delete_max` um eine Operation `decrease_all(Δ)`, die alle Prioritäten im Heap um den Wert $\Delta \geq 0$ verringert. Die Laufzeit der Operation `decrease_all` soll $\mathcal{O}(1)$ betragen. Die asymptotische worst-case Laufzeit der Operationen `insert` und `delete_max` soll durch Ihre Änderungen nicht schlechter werden.

Beschreiben Sie alle Änderungen an der Max-Heap-Datenstruktur. Beschreiben Sie außerdem den (geänderten) Ablauf der Operationen `insert` und `delete_max` sowie den Ablauf der neuen Operation `decrease_all`. Begründen Sie auch, warum die Laufzeitschranken von allen Operationen eingehalten werden.

↑↑↑ _____ / 9 Punkt(e) ↑↑↑



Gegeben sei der folgende AVL-Baum T mit dem Knoten mit Schlüssel 16 als Wurzel:



a) Geben Sie die Reihenfolge der besuchten Knoten eines *Inorder*-Durchlaufs von T an.

Reihenfolge: _____

↑↑↑ _____ / 2 Punkt(e) ↑↑↑

b) Der Schlüssel 2 wird in T eingefügt.

Geben Sie den daraus resultierenden Baum graphisch an (nachdem alle Reparatur-schritte durchgeführt wurden).

Nennen Sie auSSerdem die von Ihnen durchgeführten Reparaturschritte (Links- und Rechtsrotationen) in chronologischer Reihenfolge und in welchen Knoten diese ausgeführt wurden.

Reparaturschritte: _____

↑↑↑ _____ / 5 Punkt(e) ↑↑↑



- c) Es sei wieder der *ursprüngliche* Baum T aus der obigen Abbildung gegeben. Der Schlüssel 25 wird in T eingefügt.

Geben Sie den daraus resultierenden Baum graphisch an (nachdem alle Reparaturschritte durchgeführt wurden).

Nennen Sie außerdem die von Ihnen durchgeführten Reparaturschritte (Links- und Rechtsrotationen) in chronologischer Reihenfolge und in welchen Knoten diese ausgeführt wurden.

Reparaturschritte: _____

↑↑↑ _____ / 7 Punkt(e) ↑↑↑



a) Gegeben sei die folgende Hashtabelle der Grösse $m = 11$.

0	1	2	3	4	5	6	7	8	9	10
11			24			17	18		31	

Es wird Hashing mit linearem Austesten, $h_i(x) = (f(x)+i) \bmod 11$ (für $i = 0, 1, \dots, 10$), mit der Hashfunktion mit $f(x) = x \bmod 11$ verwendet.

Fügen Sie die folgenden Schlüssel in der gegebenen Reihenfolge in die obige Hashtabelle ein:

19, 40, 14, 34.

↑↑↑ _____ / 5 Punkt(e) ↑↑↑

b) Betrachten Sie nun eine Tabelle der Grösse $m = 5$ und Hashing mit Verkettung für die Hashfunktion $f(x) = x \bmod 5$. Es seien bereits die Schlüssel aus einer Menge M in der Hashtabelle gespeichert. Nun wird der Schlüssel 13 eingefügt.

i) (3 Punkte) Für welche der folgenden Mengen M ist die Einfügezeit von Schlüssel 13 am grössten? Kreuzen Sie genau eine Antwortmöglichkeit an.

$M := \{3, 5, 7\}$

$M := \{1, 6, 9\}$

$M := \{0, 2, 4\}$

$M := \{1, 3, 8\}$

ii) (2 Punkte) Wie gross ist der Auslastungsfaktor der Tabelle nach der Einfügung von Schlüssel 13?

↑↑↑ _____ / 5 Punkt(e) ↑↑↑



- c) Es ist das Universum $U = \{0, 1, \dots, 11\}$ gegeben. Betrachten Sie die Klasse H von Hashfunktionen

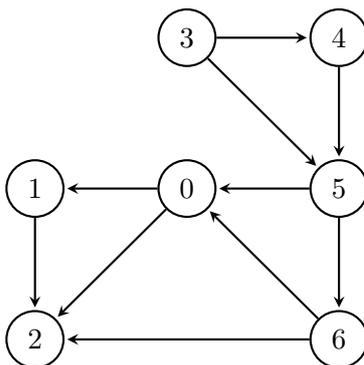
$$H = \{h_i \mid 0 \leq i < m, h_i(x) = (x + i) \bmod m\}.$$

Ist H 2-universell für eine Tabellengröße von $m = 11$? Beweisen Sie Ihre Antwort.

↑↑↑ _____ / 6 Punkt(e) ↑↑↑



Betrachten Sie den folgenden gerichteten Graphen $G = (V, E)$:



a) i) (2 Punkte) Geben Sie eine topologische Sortierung der Knoten in G an.

Topologische Sortierung: _____

ii) (2 Punkte) Existiert mehr als eine topologische Sortierung für G ? Begründen Sie Ihre Antwort kurz.

↑↑↑ _____ / 4 Punkt(e) ↑↑↑

b) Angenommen, es wird eine Breitensuche auf G ausgeführt, die im Knoten 4 beginnt. Dabei werden die Knoten in aufsteigender Reihenfolge besucht, falls ein Knoten mehr als einen Nachfolger hat. Geben Sie an, an welcher Stelle in der Besuchsreihenfolge die Knoten 0, 1, 5 sowie 6 besucht werden.

i) (1 Punkt) **Knoten 0** : An _____ter Stelle.

ii) (1 Punkt) **Knoten 1** : An _____ter Stelle.

iii) (1 Punkt) **Knoten 5** : An _____ter Stelle.

iv) (1 Punkt) **Knoten 6** : An _____ter Stelle.

↑↑↑ _____ / 4 Punkt(e) ↑↑↑



- c) i) **(6 Punkte)** Betrachten Sie nun einen Aufruf der Funktion `Tiefensuche()` aus der Vorlesung auf dem obigen Graphen G . Nehmen Sie an, dass alle Nachbarn eines Knotens stets in aufsteigender Reihenfolge in dessen Adjazenzliste stehen.

Geben Sie für jede Kante $e \in E$ von G an, ob es sich bei dieser um eine Baum- (B), Vorwärts- (V), Rückwärts- (R) oder Querkante (Q) handelt.

$e := (0, 1) : \underline{\hspace{2cm}}$ $e := (0, 2) : \underline{\hspace{2cm}}$ $e := (1, 2) : \underline{\hspace{2cm}}$

$e := (3, 4) : \underline{\hspace{2cm}}$ $e := (3, 5) : \underline{\hspace{2cm}}$ $e := (4, 5) : \underline{\hspace{2cm}}$

$e := (5, 0) : \underline{\hspace{2cm}}$ $e := (5, 6) : \underline{\hspace{2cm}}$ $e := (6, 0) : \underline{\hspace{2cm}}$

$e := (6, 2) : \underline{\hspace{2cm}}$

- ii) **(3 Punkte)** Kommen alle vier genannten Kantentypen in G vor? Geben Sie eine kurze Begründung anhand der Struktur von G .

↑↑↑ _____ / 9 Punkt(e) ↑↑↑



Alice und Bob möchten die Haustiere in der Menge $V = \{0, 1, \dots, n - 1\}$ untereinander aufteilen. Ein *Konfliktgraph* $G = (V, E)$ beschreibt, welche Tiere nicht zusammen gehalten werden können, d.h., zwischen Tier u und Tier v existiert genau dann eine Kante $\{u, v\} \in E$, wenn Tier u und Tier v sich nicht vertragen.

Ein Mengenpaar (A, B) heiSSt *Aufteilung*, wenn $A \cap B = \emptyset$ und $A \cup B = V$ gilt. Eine Aufteilung (A, B) heiSSt *gültig*, wenn jede Kante des Graphen einen Endpunkt in A und einen Endpunkt in B hat. Eine gültige Aufteilung existiert genau dann, wenn G bipartit ist. Dabei müssen die Mengen A und B nicht gleich groSS sein.

Zwei Aufteilungen (A, B) und (A', B') sind verschieden genau dann, wenn $A \neq A'$ oder $B \neq B'$ gilt. Insbesondere sind (A, B) und (B, A) verschieden.

- a) Nehmen Sie für diesen Aufgabenteil an, dass eine gültige Aufteilung existiert. Sei t die Anzahl der Zusammenhangskomponenten von G . Geben Sie die Anzahl gültiger Aufteilungen in Abhängigkeit von t an.

↑↑↑ _____ / 4 Punkt(e) ↑↑↑



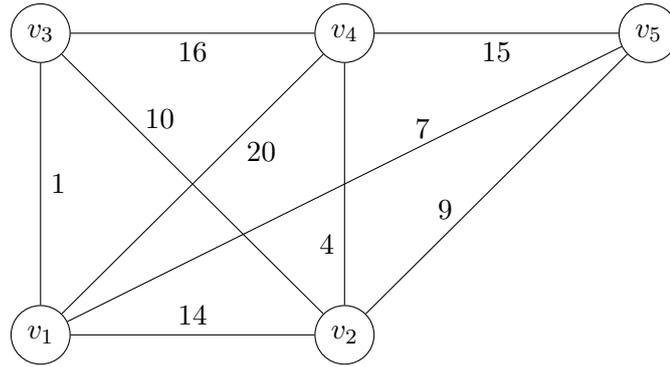
- b) Entwerfen Sie einen Algorithmus, der die Anzahl der verschiedenen gültigen Aufteilungen der Tiere berechnet. Der Konfliktgraph ist in Adjazenzlistendarstellung gegeben. Es ist nicht bekannt, ob überhaupt eine gültige Aufteilung existiert. Ihr Algorithmus soll Laufzeit $\mathcal{O}(|V| + |E|)$ haben.

Beschreiben Sie zuerst Ihre Idee und geben Sie dann den Algorithmus in Pseudo-Code. Begründen Sie auch, warum die Laufzeitschranke eingehalten wird.

↑↑↑ _____ / 9 Punkt(e) ↑↑↑



a) Gegeben sei der folgende ungerichtete Graph mit Kantengewichten:



i) (15 Punkte) Bestimmen Sie mit Hilfe des Algorithmus von Dijkstra die kürzesten Wege von Knoten v_4 zu allen anderen Knoten. Füllen Sie hierzu die nachfolgende Tabelle aus, wobei jede Zeile einem Schritt des Algorithmus entspricht.

Die Spalte d soll den aktuellen Distanzwert des Knoten enthalten, die Spalte p den aktuellen Vorgänger des Knoten im Elternarray.

Menge S	v_1		v_2		v_3		v_4		v_5	
	d	p								



ii) (4 Punkte) Zeichnen Sie den resultierenden Baum kürzester Wege:



↑↑↑ _____ / 19 Punkt(e) ↑↑↑

b) Sei $G = (V, E)$ nun ein beliebiger gewichteter ungerichteter Graph. Wir transformieren die Kantengewichte von G : Das Gewicht $w(e)$ einer Kante $e \in E$ ändert sich zu $w'(e)$.

In welchem der folgenden Fälle werden sich die kürzesten Wege in der Transformation niemals ändern? Kreuzen Sie genau eine Antwortmöglichkeit an.

$w'(e) = 2 \cdot w(e) + 1$

$w'(e) = w(e) \cdot w(e)$

$w'(e) = \log_2(|V| + 2) \cdot w(e)$

$w'(e) = \log_2(w(e))$

↑↑↑ _____ / 3 Punkt(e) ↑↑↑



- a) Gegeben seien $n > 0$ Zeichen z_1, z_2, \dots, z_n , wobei Zeichen z_i die absolute Häufigkeit $H(z_i) = 2^{i-1}$ hat, für $i \in \{1, \dots, n\}$.

Geben Sie die Tiefe des zugehörigen Huffman-Baums asymptotisch exakt an. Begründen Sie Ihre Antwort kurz.

Tiefe: _____

Kurze Begründung:

↑↑↑ _____ / 4 Punkt(e) ↑↑↑

- b) Bei der *Union-Find*-Datenstruktur wird in jedem Union-Schritt stets die Wurzel des „kleineren“ Baums unter die Wurzel des „grösseren“ Baums gehängt.

Für welche der folgenden Definitionen von „GrösSe“ eines Baums B kann die Laufzeit einer `find`-Operation asymptotisch linear in der Anzahl der Elemente in der Union-Find-Datenstruktur werden? Kreuzen Sie genau eine Antwortmöglichkeit an.

Wenn die „GrösSe“ von B definiert wird als...

- ...die Knotenzahl von B .
- ...der durchschnittliche Knotengrad von B .
- ...der grösSte in B vorkommende Knotenindex.
- ...die Kantenanzahl von B .

↑↑↑ _____ / 3 Punkt(e) ↑↑↑



c) Gegeben sei ein gewichteter gerichteter Graph $G = (V, E)$ mit Knotenmenge $V = \{1, 2, \dots, n\}$ und Kantenmenge $E = \{(i, i + 1) \mid 1 \leq i < n\}$. Geben Sie die exakte asymptotische Laufzeit folgender Algorithmen auf G in Abhängigkeit von n an.

i) (2 Punkte) Dijkstras Algorithmus mit Startknoten $s = 1$:

$\Theta(\text{_____})$

ii) (2 Punkte) Algorithmus von Floyd:

$\Theta(\text{_____})$

↑↑↑ _____ / 4 Punkt(e) ↑↑↑

d) Im Folgenden wird *Stabiles Matching* für zwei Mengen A und B mit $|A| = |B| = n$ betrachtet. Die Agenten in A sind aktiv und machen Anträge. Alle Agenten in einer Menge haben eine vollständige Präferenzordnung über die Agenten in der jeweils anderen Menge.

Geben Sie an, nach wie vielen Anträgen der Deferred-Acceptance Algorithmus bei den folgenden Präferenzordnungen terminiert.

i) (2 Punkte) Alle Agenten in A haben voneinander verschiedene Agenten aus B an der ersten Stelle ihrer Präferenzordnung, d.h., denselben Favoriten.

ii) (2 Punkte) Alle Agenten in A haben eine identische Präferenzordnung.

↑↑↑ _____ / 4 Punkt(e) ↑↑↑



Ein Gebiet ist in Planquadrate $(x, y) \in \{0, 1, \dots, n\}^2$ aufgeteilt. Jedes Planquadrat (x, y) hat eine Höhe $h(x, y)$ in Metern über dem Meeresspiegel.

Ein Wanderer möchte von Planquadrat $(0, 0)$ zu (n, n) laufen und dabei möglichst wenige Höhenmeter überwinden. Die Wahl seines Weges unterliegt dabei folgenden Einschränkungen: Befindet er sich in einem Planquadrat (x, y) , kann er entweder zum nächsten Planquadrat $(x+1, y)$ in x -Richtung oder zum nächsten Planquadrat $(x, y+1)$ in y -Richtung laufen, sofern er dabei das Gebiet nicht verlässt.

Die Höhendifferenz eines Weges $(x_0, y_0) \rightarrow (x_1, y_1) \rightarrow (x_2, y_2) \rightarrow \dots \rightarrow (x_k, y_k)$ von Planquadrat (x_0, y_0) zu Planquadrat (x_k, y_k) ist gegeben durch $\sum_{\ell=1}^k |h(x_\ell, y_\ell) - h(x_{\ell-1}, y_{\ell-1})|$.

Um die minimale Höhendifferenz eines Weges von $(0, 0)$ nach (n, n) zu berechnen, werden Teilprobleme für $x, y \in \{0, 1, \dots, n\}$ wie folgt definiert:

$\text{opt}(x, y) =$ minimale Höhendifferenz, die überwunden werden muss, um von $(0, 0)$ zu (x, y) zu gelangen.

- a) Geben Sie eine Rekursionsgleichung sowie Basisfälle für obige Teilprobleme an.

Basisfälle:

Rekursionsgleichung:

↑↑↑ _____ / 7 Punkt(e) ↑↑↑



- b) Entwickeln Sie aus Ihrer Rekursionsgleichung ein dynamisches Programm zur Berechnung der minimal nötigen Höhendifferenz. Ihr Algorithmus soll Laufzeit $\mathcal{O}(n^2)$ haben. Geben Sie Ihren Algorithmus in Pseudo-Code an und analysieren Sie dessen Laufzeit.

↑↑↑ _____ / 7 Punkt(e) ↑↑↑



Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit ganzzahligen Kantengewichten aus der Menge $\{1, 2, \dots, k\}$, wobei k eine *Konstante* unabhängig von der Grösse des Graphen ist.

Entwerfen Sie einen Algorithmus, der in Zeit $\mathcal{O}(|V| + |E|)$ einen minimalen Spannbaum von G berechnet. G ist in Adjazenzlistendarstellung gegeben.

Beschreiben Sie Ihren Algorithmus in Worten und begründen Sie, warum die gegebene Laufzeitschranke eingehalten wird.



Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Hinweis auf Seite 23 platziert wurde.

