

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.**

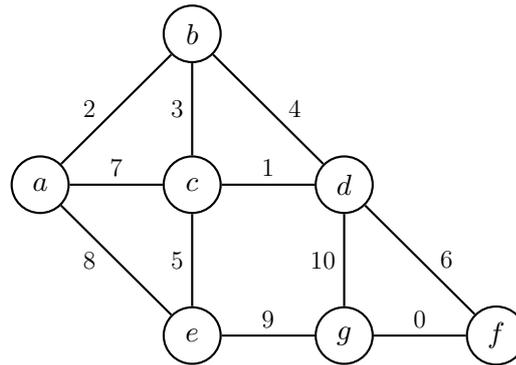


**Diese Seite ist nur für den internen Gebrauch bestimmt.  
Bitte nicht beschreiben.**

Aufgabe	1	2	3	4	5	6	7	8	9	10	11
Erreichbar	16	22	14	17	17	16	9	10	14	13	12
Erreicht											

Klausur	Bonifikation

a) Gegeben sei der folgend dargestellte ungerichtete, gewichtete Graph  $G$ :



Falls benötigt, sei  $a$  der Startknoten. Für die beiden folgenden Teilaufgaben i), ii) und iii) ist die Reihenfolge, in der gleichwertige Kanten/Knoten ausgewählt werden, beliebig.

i) **(5 Punkte)** Welche Kante wird vom DJP-Algorithmus als *viertes* in den minimalen Spannbaum von  $G$  eingefügt?

**Antwort:** \_\_\_\_\_

ii) **(5 Punkte)** Welche Kante wird von Kruskals Algorithmus als erste *nicht* in den minimalen Spannbaum von  $G$  eingefügt?

**Antwort:** \_\_\_\_\_

iii) **(3 Punkte)** Geben Sie die maximale Tiefe eines Baums in der Union-Find Datenstruktur bei der Anwendung von Kruskals Algorithmus auf dieser Instanz an. Hierbei soll in jeder Union-Operation die Wurzel des kleineren Baums unter die Wurzel des grösseren Baums gehängt werden. Dabei ist der Baum mit mehr Knoten der grössere. Bei gleicher Knotenzahl ist der Baum mit lexikographisch kleinerer Wurzel der grössere.

**Antwort:** \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 13 Punkt(e) ↑↑↑



- b) Gegeben ist ein ungerichteter, zusammenhängender Graph, der kein Baum ist. Ist es möglich, dass der DJP-Algorithmus die schwerste im minimalen Spannbaum enthaltene Kante als erstes hinzufügt? Geben Sie ein begründetes Beispiel an oder begründen Sie, warum das nicht möglich ist.

↑↑↑ \_\_\_\_\_ / 3 Punkt(e) ↑↑↑





- c) Geben Sie für die drei folgenden Algorithmen in Pseudo-Code jeweils die Laufzeit in Abhängigkeit von  $n$  in  $\Theta$ -Notation an.

```

for  $i = 1 \dots n$  do
   $j = 2n$ ;
  while  $j \geq i$  do
     $j = j - 2$ ;

```

$\Theta(\text{_____})$

```

if  $n \leq 2022$  then
   $i = 1$ ;  $j = n$ ;
  while  $i < j$  do
     $i = 2 * i$ ;
     $j = j - 1$ ;

```

$\Theta(\text{_____})$

```

for  $i = 1 \dots n$  do
   $j = 1$ ;
  while  $j < i$  do
     $j = 2 * j$ ;

```

$\Theta(\text{_____})$

↑↑↑ \_\_\_\_\_ / 6 Punkt(e) ↑↑↑

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

```

rek(int i, int j) {
  if( $i \geq j$ ) return;
  int h = A[i];
  A[i] = A[j];
  A[j] = h;
  rek(i+1, j-1);
}

```

- i) (2 Punkte) Beschreiben Sie kurz, welche Modifikation durch den Aufruf `rek(1, n)` am Array  $A$  vorgenommen wird.

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

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

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

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

$T(n) = T(n - 2) + O(1)$

↑↑↑ \_\_\_\_\_ / 4 Punkt(e) ↑↑↑



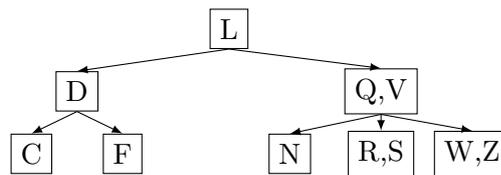
- a) Sei  $T$  ein  $(a, b)$ -Baum mit  $b = 2a - 1$ , also ein B-Baum. Die Tiefe von  $T$  habe sich nach Entfernen eines Schlüssels von 2 auf 1 gesenkt.

Wie viele Schlüssel befinden sich nach dem Entfernen *mindestens* im Baum? Geben Sie eine exakte untere Schranke in Abhängigkeit von  $a$  an.

**Antwort:** \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 5 Punkt(e) ↑↑↑

- b) Betrachten Sie den folgenden (2,3)-Baum:

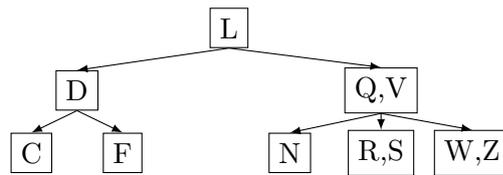


Führen Sie die Operation `remove(C)` aus. Geben Sie den resultierenden (2,3)-Baum in graphischer Darstellung an.

↑↑↑ \_\_\_\_\_ / 5 Punkt(e) ↑↑↑



c) Betrachten Sie den folgenden (2,3)-Baum:

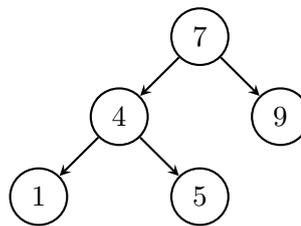


Führen Sie die Operation `insert(T)` aus. Geben Sie den resultierenden (2,3)-Baum in graphischer Darstellung an.

↑↑↑ \_\_\_\_\_ / 4 Punkt(e) ↑↑↑



- a) Wie viele verschiedene Folgen von Einfügeoperationen führen zu dem folgenden binären Suchbaum? Gehen Sie davon aus, dass der Suchbaum am Anfang leer ist und jeder Schlüsselwert genau einmal vorliegt.



**Antwort:** \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 3 Punkt(e) ↑↑↑

- b) Sei das Array  $A = [9, 5, 3, 1, 4]$  ein *Max-Heap*. Geben Sie  $A$  nach dem Einfügen der Priorität 12 an.

**Antwort:**  $A = [$  \_\_\_\_\_  $]$

↑↑↑ \_\_\_\_\_ / 3 Punkt(e) ↑↑↑

- c) Die Prioritäten  $1, \dots, n$  werden mittels *insert* in folgender Reihenfolge in einen anfangs leeren *Max-Heap* eingefügt:  $n, n - 1, n - 2, \dots, 3, 2, 1$ . Welche asymptotische Gesamtlaufzeit ergibt sich für die  $n$  Einfügeoperationen in diesem Fall?

$\Theta$ (\_\_\_\_\_)

↑↑↑ \_\_\_\_\_ / 3 Punkt(e) ↑↑↑

- d) Ein binärer Baum wird als *vollständig* bezeichnet, wenn alle Blätter die gleiche Tiefe haben und jeder innere Knoten genau zwei Kinder besitzt.

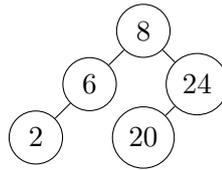
Sei  $B$  ein vollständiger binärer Suchbaum mit  $2^k - 1$  ganzzahligen und paarweise verschiedenen Schlüssel aus dem Intervall  $[1, 2^k]$ . Welche Schlüssel können an der Wurzel von  $B$  gespeichert sein?

**Antwort:** \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 3 Punkt(e) ↑↑↑



e) Gegeben sei der folgende AVL-Baum:



Fügen Sie einen geeigneten ganzzahligen Schlüssel  $x > 0$  ein, so dass der resultierende Baum die AVL-Eigenschaft direkt behält und keine Reorganisationen bzw. Reparaturen notwendig sind. Zeichnen Sie den resultierenden Baum.

**Schlüssel**  $x =$  \_\_\_\_\_

**Resultierender Baum:**

↑↑↑ \_\_\_\_\_ / 5 Punkt(e) ↑↑↑



- a) Sei  $G = (V, E)$  ein beliebiger ungerichteter Graph mit strikt positiven Kantengewichten. Wir transformieren die Gewichte von  $G$ : Das Gewicht  $w(e)$  einer Kante  $e \in E$  ändert sich zu  $w'(e)$ .

In welchem der folgenden Fälle bleibt ein minimaler Spannbaum durch die Transformation *nicht immer* minimal? Kreuzen Sie genau eine Antwort an.

$w'(e) = (w(e) - 2)^2$

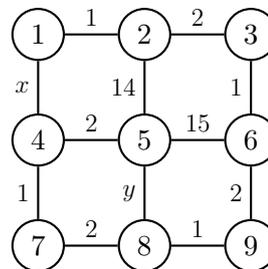
$w'(e) = 3 \cdot (w(e) + 1)$

$w'(e) = \sqrt{w(e)}$

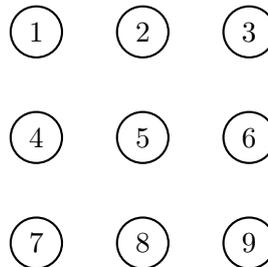
$w'(e) = \log_2(w(e))$

↑↑↑ \_\_\_\_\_ / 4 Punkt(e) ↑↑↑

- b) Gegeben sei der folgende Graph  $G$  mit variablen Kantengewichten  $x, y \geq 0$ :



- i) (4 Punkte) Bestimmen Sie einen Baum kürzester Wege ausgehend von Knoten 1 für  $x = 12$  und  $y = 6$ . Nutzen Sie folgende Vorlage:



- ii) (3 Punkte) Sei  $D \subseteq \mathbb{R}^2$  die Menge *aller* Paare  $(x, y)$ , so dass der Baum aus i) weiterhin ein Baum kürzester Wege für Startknoten 1 ist. Kreuzen Sie die richtige Antwort an:

$D = \{(x, y) \subseteq \mathbb{R}^2 \mid x \geq 99, y \geq 99\}$

$D = \{(x, y) \subseteq \mathbb{R}^2 \mid x \geq 12, y \geq 6\}$

$D = \{(x, y) \subseteq \mathbb{R}^2 \mid x \leq 10, y \leq 5\}$

$D = \{(x, y) \subseteq \mathbb{R}^2 \mid x \geq 10, y \geq 5\}$

↑↑↑ \_\_\_\_\_ / 7 Punkt(e) ↑↑↑



- c) Im Folgenden wird *Stabiles Matching* für zwei Mengen  $A = \{a_1, a_2\}$  und  $B = \{b_1, b_2\}$  betrachtet.

Gegeben seien folgende Präferenzlisten:

$a_1 : b_1 \succ b_2$

$b_1 : a_2 \succ a_1$

$a_2 : b_2 \succ b_1$

$b_2 : a_1 \succ a_2$

Geben Sie *alle* stabilen Matchings für obige Instanz an.

**Antwort:** \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 4 Punkt(e) ↑↑↑

- d) Ist  $a \rightarrow 1, b \rightarrow 0, c \rightarrow 11, d \rightarrow 10$  ein Huffman-Code für folgenden Text: "aaabaabacd" ?

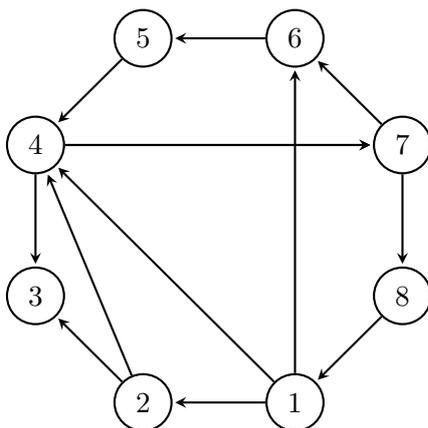
Ja

Nein

↑↑↑ \_\_\_\_\_ / 2 Punkt(e) ↑↑↑



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



- a) i) (2 Punkte) Welche Kante  $e \in E$  muss aus  $G$  entfernt werden, damit eine topologische Sortierung *aller* Knoten des Graphen  $G' = (V, E')$  mit  $E' = E \setminus \{e\}$  möglich ist?

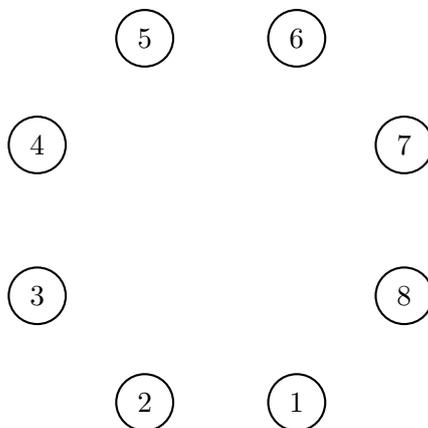
Antwort: \_\_\_\_\_

- ii) (2 Punkte) Geben Sie eine entsprechende topologische Sortierung der Knoten in  $G'$  an. Falls in einem Schritt mehrere Knoten gewählt werden können, wählen Sie den kleinsten.

Antwort: \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 4 Punkt(e) ↑↑↑

- b) i) (4 Punkte) Bestimmen Sie den Baum der Tiefensuche für  $G$ , wie in der obigen Abbildung gegeben, wenn die Tiefensuche in Knoten 1 gestartet wird. Dabei sollen die unbesuchten Nachbarn eines jeden Knotens in aufsteigender Reihenfolge besucht werden. Sie können die folgende Vorlage verwenden.



ii) (5 Punkte) Geben Sie alle Quer-, Rückwärts, und Vorwärtskanten an.

Querkanten: \_\_\_\_\_

Rückwärtskanten: \_\_\_\_\_

Vorwärtskanten: \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 9 Punkt(e) ↑↑↑

c) Auf einem gerichteten, stark zusammenhängenden Graphen  $G = (V, E)$  wird eine Tiefensuche ausgeführt. Geben Sie die Summe der Anzahl aller Vorwärts-, Rückwärts-, und Querkanten in Abhängigkeit von  $|E|$  und  $|V|$  an.

Antwort: \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 3 Punkt(e) ↑↑↑



Gegeben sei ein Min-Heap  $H$  mit  $n$  Zahlen. Erweitern Sie die Heap-Datenstruktur um eine Operation `smaller(x)`, die alle Zahlen in  $H$  ausgibt, die strikt kleiner als  $x$  sind. Die Laufzeit soll  $\mathcal{O}(k)$  sein, wobei  $k$  die Anzahl der Elemente in  $H$  ist, die strikt kleiner als  $x$  sind.

Geben Sie Ihr Verfahren in Pseudo-Code an und begründen Sie, warum die Laufzeitschranke eingehalten wird.



a) Gegeben sei die folgende Hashtabelle der Grösse  $m = 7$ .

0	1	2	3	4	5	6
70	22	51			47	

Es wird Hashing mit linearem Austesten,  $h_i(x) = (f(x) + i) \bmod 7$  (für  $i = 0, 1, \dots, 6$ ), mit der Hashfunktion  $f(x) = x \bmod 7$  verwendet.

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

64, 39, 11.

↑↑↑ \_\_\_\_\_ / 6 Punkt(e) ↑↑↑

b) Sei nun die Tabellengrösse  $m = 11$  und das Universum  $U = \{0, 1, 2, \dots, 28\}$ . Welche der folgenden Klassen von Hashfunktionen ist  $c$ -universell für das angegebene  $c$ ? Kreuzen Sie genau eine Antwort an.

$H = \{h_i \mid 0 \leq i < 11, h_i(x) = (x + i) \bmod 11\}, c = 2$

$H = \{h_{a,b} \mid 0 \leq a, b < 29, h_{a,b}(x) = ((ax + b) \bmod 29) \bmod 11\}, c = \frac{3}{2}$

$H = \{h_i \mid 0 \leq i < 11, h_i(x) = (i \cdot x) \bmod 11\}, c = 2$

$H = \{h_{a,b} \mid 0 \leq a, b < 29, h_{a,b}(x) = (ax + b) \bmod 11\}, c = 1$

↑↑↑ \_\_\_\_\_ / 4 Punkt(e) ↑↑↑



Entlang einer *EinbahnstraSSe* befinden sich die Bushaltestellen  $0, 1, \dots, n$ . An *jeder* Haltestelle kann entweder das Ticket WEIT oder das Ticket NAH gekauft werden. Die Preise für die Tickets sind jeweils abhängig von der Haltestelle, wobei Ticket WEIT auch günstiger sein kann als Ticket NAH. An Haltestelle  $i$  kostet das Ticket WEIT  $W(i)$  Euro und das Ticket NAH  $N(i)$  Euro. Mit dem Ticket WEIT kann bis zu *zwölf* Haltestellen weiter gefahren werden, mit dem Ticket NAH nur *eine*.

Ziel ist es, mit möglichst geringen Kosten von Haltestelle 0 zu Haltestelle  $n$  mit dem Bus zu fahren.

Um das Problem zu lösen, werden Teilprobleme für  $i \in \{0, 1, \dots, n\}$  wie folgt definiert:

$\text{opt}(i)$  = minimale Kosten, um von Haltestelle 0  
zu Haltestelle  $i$  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 minimalen Kosten, um von Haltestelle 0 zu Haltestelle  $n$  mit dem Bus zu fahren. Ihr Algorithmus soll Laufzeit  $\mathcal{O}(n)$  haben.

Geben Sie Ihren Algorithmus in Pseudo-Code an und analysieren Sie dessen Laufzeit.

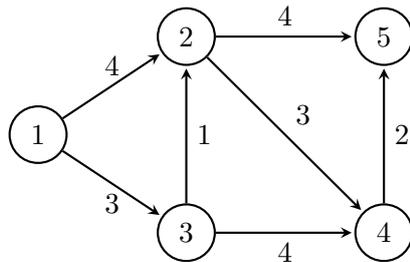
↑↑↑ \_\_\_\_\_ / 7 Punkt(e) ↑↑↑



Im Problem *Single-Source-Min-Max-Paths (SSMMP)* ist ein gerichteter Graph  $G = (V, E)$  und ein Startknoten  $s \in V$  gegeben. Jede Kante  $e \in E$  hat ein Gewicht  $w(e) > 0$ . Gesucht ist die *Min-Max-Distanz* von Knoten  $s$  zu allen anderen Knoten  $v \neq s$ . Die Min-Max-Distanz  $MMD(u, v)$  von Knoten  $u$  zu Knoten  $v$  ist dabei definiert als

$$MMD(u, v) = \min_{P \text{ ist Pfad von } u \text{ nach } v} \max_{e \text{ ist Kante des Pfades } P} w(e)$$

a) Geben Sie die Min-Max-Distanz von Knoten 1 zu Knoten 5 für folgenden Graphen an:



$MMD(1, 5) = \underline{\hspace{2cm}}$

↑↑↑ \_\_\_\_\_ / 4 Punkt(e) ↑↑↑

b) Entwerfen Sie einen möglichst effizienten Algorithmus, der das Problem für beliebige Instanzen löst. Sie können davon ausgehen, dass der Graph in Adjazenzlistendarstellung gegeben ist. Ausgabe Ihres Algorithmus sollen die Min-Max-Distanzen von  $s$  zu allen anderen Knoten sein.

Beschreiben Sie Ihren Algorithmus in Worten und analysieren Sie seine worst-case Laufzeit. Begründen Sie, warum Ihr Verfahren funktioniert.

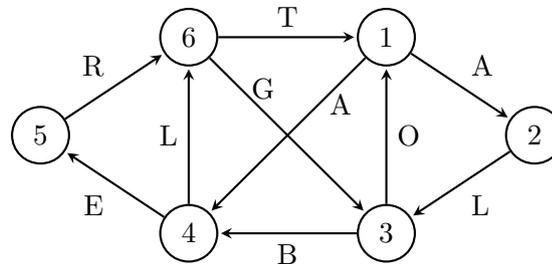
↑↑↑ \_\_\_\_\_ / 9 Punkt(e) ↑↑↑



Gegeben sei ein gerichteter Graph  $G = (V, E)$  in Adjazenzlistendarstellung. Jede Kante des Graphen hat als Beschriftung einen Großbuchstaben.

Ziel ist es, einen Weg der Länge vier zu finden, dessen Kantenbeschriftungen mit der Zeichenkette ALGO übereinstimmen, oder zu verifizieren, dass es keinen solchen Weg gibt.

a) Betrachten Sie folgende Instanz des Problems:



Geben Sie einen Weg an, dessen Kantenbeschriftungen mit der Zeichenkette ALGO übereinstimmen.

**Antwort:** \_\_\_\_\_

↑↑↑ \_\_\_\_\_ / 3 Punkt(e) ↑↑↑



- b) Entwerfen Sie einen Algorithmus, der entscheidet, ob in einem gegebenen Graphen  $G$  ein zum String **ALGO** passender Weg existiert oder nicht. Die asymptotische Laufzeit soll  $\mathcal{O}(|V| + |E|)$  nicht übersteigen.

Beschreiben Sie Ihren Algorithmus in Worten. Begründen Sie auch, warum die Laufzeitschranke eingehalten wird.

↑↑↑ \_\_\_\_\_ / 9 Punkt(e) ↑↑↑



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



**Wichtig:** Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Hinweis auf Seite 24 platziert wurde.

