TU Wien
Institut für Logic and Computation
Algorithms and Complexity Group

186.866 Algorithmen und Datenstrukturen VU 8.0

Nachtragstest, 2021S

28. September 2021

Gruppe D

Es gelten die in TUWEL verkündeten Regelungen.

Füllen Sie nachfolgende Felder des Deckblattes aus bzw. fertigen Sie ein eigenes Deckblatt mit diesen Informationen an. Fügen Sie das ausgefüllte Deckblatt als erste Seite von Ihrem PDF mit den Lösungen hinzu.

Aufgabe D1: Algorithmenanalyse, P und NP **(20 Punkte)**

a) (12 Punkte) Bestimmen Sie die Laufzeiten der angegebenen Algorithmen in Abhängigkeit vom Eingabeparameter n in Θ-Notation. Verwenden Sie hierfür möglichst einfache Terme.

(i)

```
a ← 1
for i = 1, ..., n
    for j = 1, ..., 4
        a ← 2 * a
for i = −15, ..., n√n
    a ← √(a^3)
return a
```

Laufzeit (in Θ-Notation):

(ii)

```
b ← 2
for i = 1, ..., n − 3
    for j = i, ..., n + 5
        b ← b + i
return b
```

Laufzeit (in Θ-Notation):

(iii)

```
c ← 1
for i = 1, ..., √n
    k ← 1
    while k < n + i
        c ← c + i
        k ← 5 * k
return c
```

Laufzeit (in Θ-Notation):

b) (2 Punkte) Nehmen Sie an, wir haben für ein $n_0 \in \mathbb{N}^+$ gezeigt, dass für alle $n \geq n_0$ gilt

$$n + \log 3n \leq 45 \cdot n.$$

Was haben wir somit gezeigt? Kreuzen/Geben Sie alle zutreffenden Aussagen an:

(MC1) ☐ $n + \log 3n$ ist in $O(n)$

(MC2) ☐ $n + \log 3n$ ist in $\Theta(n)$

(MC3) ☐ $n + \log 3n$ ist in $\Omega(n)$

(MC4) ☐ keine der zuvor genannten Aussagen

c) (6 Punkte) Seien A, B, C Ja/Nein-Probleme und n die Eingabegröße. Nehmen Sie an, es gibt

- eine Reduktion von A nach B in Zeit n^2,
- eine Reduktion von B nach C in Zeit n,
- eine Reduktion von A nach C in Zeit n^3,
- eine Reduktion von C nach HAM-CYCLE in Zeit $n \log n$,

Geben Sie für die folgenden Aussagen an, ob diese unter den obigen Annahmen sicher wahr (w), sicher falsch (f), oder keine Aussage möglich ist, wir bezeichnen das dann als unbestimmt (u).

(Q1) Es gibt eine Polynomialzeitreduktion von B nach HAM-CYCLE.

☒ Wahr ☐ Falsch ☐ unbestimmt

(Q2) C ist ein NP-vollständiges Problem.

☒ Wahr ☐ Falsch ☐ unbestimmt

(Q3) Wenn C in $O(n\sqrt{n})$ gelöst werden kann dann kann B in $O(n^2)$ gelöst werden.

☐ Wahr ☐ Falsch ☒ unbestimmt

(20 Punkte)

Aufgabe D2: Graphen

a) (4 Punkte) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind. *(+1 Punkt für jede richtige, -1 Punkt für jede falsche und 0 Punkte für keine Antwort, keine negativen Punkte)*

(Q1) Jeder ungerichtete Graph mit 10 Knoten, in dem jeder Knoten den Knotengrad 3 hat, enthält genau 15 Kanten.

☒ Wahr ☐ Falsch

(Q2) Jeder schwach zusammenhängende Graph mit n Knoten hat mindestens $n-1$ Kanten.

☒ Wahr ☐ Falsch

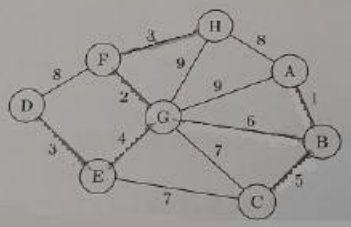
(Q3) Jeder ungerichtete Graph mit n Knoten und mindestens n Kanten enthält einen Kreis.

☒ Wahr ☐ Falsch

(Q4) Ein gerichteter Graph ist genau dann azyklisch, wenn er eine topologische Sortierung hat.

☐ Wahr ☐ Falsch

b) (6 Punkte) Berechnen Sie einen minimalen Spannbaum des untenstehenden Graphen. Geben Sie eine mögliche Reihenfolge an, in der der Algorithmus von **Kruskal** die Kanten des Graphen in einen Spannbaum einfügt. Geben Sie die Reihenfolge als Liste von Knotenpaaren an (z.B.: AB, BC, CD, ...).



c) *(2 Punkte)* Hat der obige Graph einen eindeutigen minimalen Spannbaum? *Begründen Sie Ihre Antwort.*

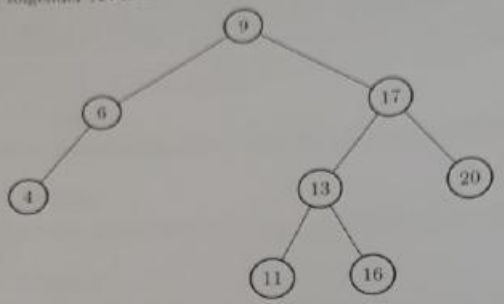
d) (8 Punkte) Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit Knotenmenge $V = \{1, \ldots, n\}$ und m Kanten.

Geben Sie Pseudocode für einen Algorithmus mit Laufzeit $O(n + m)$ an, der die (schwachen) Zusammenhangskomponenten von G bestimmt. Ihr Algorithmus soll ein Array `component` der Länge n ausgeben, sodass `component`$[v] =$ `component`$[w]$ genau dann gilt, wenn die Knoten v und w in einer Zusammenhangskomponente liegen.

(20 Punkte)

Aufgabe D3: AVL-Bäume und Hashing

a) Gegeben ist folgender **AVL-Baum**



(i) (6 Punkte) Fügen Sie die Schlüssel 10 und 14 in dieser Reihenfolge in diesen AVL-Baum ein. Falls notwendig, so rebalancieren Sie den Baum nach jedem Einfügen mit geeigneten Rotationsoperationen, um wieder einen gültigen AVL-Baum zu erhalten. Zeichnen Sie den vollständigen Baum direkt nach dem Einfügen und nach jedem Rotationsschritt.

(ii) (6 Punkte) Löschen Sie aus dem **ursprünglichen, gegebenen** AVL-Baum die Schlüssel 11 und 20 in dieser Reihenfolge. Falls notwendig rebalancieren Sie den Baum nach jedem Löschvorgang mit geeigneten Rotationsoperationen. Zeichnen Sie wiederum den vollständigen Baum direkt nach dem Löschen und nach jedem Rotationsschritt.

b) (4 Punkte) Gegeben sind folgende natürliche Zahlen:

$$[27, 12, 8, 15]$$

Fügen Sie diese in der vorgegebenen Reihenfolge in eine anfangs leere Hashtabelle der Größe $m = 6$ mit der Hashfunktion $h(k) = k \bmod m$ ein. Kollisionen sollen mit **Verkettung der Überläufer** behandelt werden.

Berechnen Sie anschließend die durchschnittliche Anzahl T von Schlüsselvergleichen bei einer erfolgreichen Suche eines Elements, wobei jedes dieser Elemente mit gleicher Wahrscheinlichkeit gesucht wird.

Hashtabelle:

0	1	2	3	4	5

$T =$

c) Gegeben ist eine Hashtabelle der Größe $m = 8$ mit Hashfunktion $h(k) = k \bmod m$. Als Kollisionsbehandlung wird **Lineares Sondieren** verwendet.

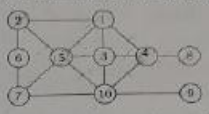
0	1	2	3	4	5	6	7
0	15		11			22	6

(i) (2 Punkte) Geben Sie eine Reihenfolge an, in der die enthaltenen Elemente in die anfangs leere Hashtabelle eingefügt worden sein könnten, um diesen Zustand ohne irgendwelche anderen Schritte zu erreichen.

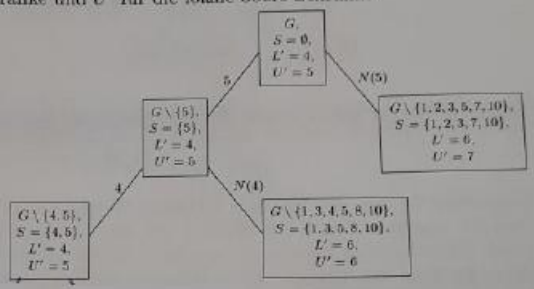
(ii) (2 Punkte) Mit welcher Wahrscheinlichkeit wird eine neu einzufügende zufällige positive ganze Zahl an die Position 2 gesetzt? Begründen Sie Ihre Antwort.

Aufgabe D4: Branch and Bound, Dynamische Programmierung (20 Punkte)

a) (10 Punkte) Auf dem folgenden Graphen soll mittels des Branch and Bound Algorithmus aus der Vorlesung ein minimales Vertex Cover gefunden werden.



Betrachten Sie die folgende Skizze eines partiellen Ablaufs des Branch and Bound Algorithmus. In der Skizze geben die ersten beiden Zeilen das aktuelle Teilproblem an. In der ersten Zeile wird der Graph des aktuellen Teilproblems angegeben, in der zweiten Zeile die aktuelle partielle Lösung S. Weiter steht L' für die lokale untere Schranke und U' für die lokale obere Schranke.



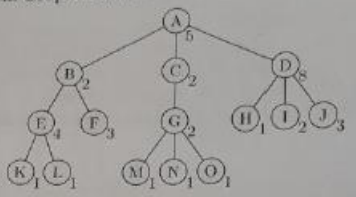
(i) Vervollständigen Sie die Skizze auf der vorigen Seite, sodass sie einen vollständigen Ablauf des Branch and Bound Algorithmus darstellt.

(ii) Geben Sie ein maximales Matching, das die untere Schranke 4 für den Graphen $G \setminus \{4, 5\}$ rechtfertigt, an

b) (10 Punkte) Das Minimum-Weight Vertex Cover Problem ist eine Version des Vertex Cover Problems für gewichtete Graphen, das wie folgt definiert ist:

Gegeben sei ein schlichter Graph $G = (V, E)$ mit positiven Knotengewichten $w_v \in \mathbb{R}^+$ für alle Knoten $v \in V$. Gesucht ist ein Vertex Cover $V' \subset V$ mit minimalem Gewicht, d.h. für jede Kante $(u, v) \in E$ gilt $\{u, v\} \cap V' \neq \emptyset$ und $\sum_{v \in V'} w_v$ ist minimal über alle möglichen Vertex Cover von G.

(i) Geben Sie für den nachfolgenden Graphen ein Vertex Cover mit minimalem Gewicht an. In dem Graphen sind die Gewichte neben den Knoten angegeben.



Minimum- Weight Vertex Cover:

(ii) Betrachten Sie nun den Fall, des Minimum-Weight Vertex Cover Problems, dass der gegebene Graph ein Baum T ist. Im Folgenden ist ein Gerüst für einen Algorithmus in Pseudocode gegeben, der das Minimum-Weight Vertex Cover Problem auf Bäumen mittels dynamischer Programmierung lösen soll. Für jeden Knoten $u \in V$ (mit Gewicht w_u) sollen die Tabelleneinträge $M_{\text{in}}[u]$ und $M_{\text{out}}[u]$ ausgefüllt werden. Am Ende enthält $M_{\text{in}}[u]$ den besten Wert für eine Lösung des Teilbaums T_u mit u als Wurzel, sodass u Teil der Lösung ist. Äquivalent enthält $M_{out}[u]$ den besten Wert einer Lösung für T_u, aber unter der Annahme, dass u *nicht* Teil der Lösung ist.

Am Ende soll der Algorithmus das minimale Gewicht eines Vertex Cover von T retournieren. Ergänzen Sie die markierten Zeilen so, dass der Algorithmus korrekt ist.

`TreeMinWeightVC(`T, *Wurzel* r`)`:
foreach u *in* T *in Postorder* **do**
  if u *ist ein Blatt* **then**
    $M_{\text{in}}[u] = w_u$
    $M_{\text{out}}[u] = 0$
  else
    $M_{\text{in}}[u] \leftarrow$
    $M_{\text{out}}[u] \leftarrow$
return $\min\{M_{in}[r], M_{out}[r]\}$

Aufgabe D5: Approximation und Heuristische Verfahren **(20 Punkte)**

Wir betrachen das aus der Vorlesung bekannte MAX-SAT Problem, bei dem eine Boole'sche Formel in konjunktiver Normalform gegeben ist und nach einer Variablenzuweisung gesucht wird, welche die Anzahl der erfüllten Klauseln in der gegebenen Formel maximiert.

a) (8 Punkte) Angenommen A und B sind Approximationsalgorithmen für MAX-SAT, A mit einer Gütegarantie von $\frac{3}{4}$ und B mit einer Gütegarantie von 1.

(i) Gegeben ist eine Formel F mit 60 Klauseln. Es gibt eine Variablenzuweisung die 48 Klauseln in F erfüllt. Es gibt allerdings keine Variablenzuweisung die mehr als 48 Klauseln in F erfüllt. Weiters ist I eine Variablenzuweisung die mittels A für F gefunden wurde. In welchem Wertebereich befindet sich die Anzahl der von I erfüllten Klauseln?

(ii) Angenommen A findet für eine andere Formel F' eine Variablenzuweisung die 150 Klauseln in F' erfüllt. In welchem Wertebereich befindet sich eine optimale Lösung für F'?

(iii) Wenn B ein Polynomialzeitalgorithmus ist, was kann dann über die Komplexitätsklassen P und NP ausgesagt werden? Gilt dann $P = NP$, $P \neq NP$ oder kann keine Aussage getroffen werden? Begründen Sie Ihre Antwort kurz. *Hinweis: Überlegen Sie, ob sich so mittels B das allgemeine* SAT*-Problem in polynomieller Zeit lösen lässt.*

b) (12 Punkte) Sei G die folgende Boole'sche Formel:

$$G = (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_4} \vee x_1).$$

Variablenzuweisungen werden als binäre Vektoren $\langle x_1, x_2, x_3, x_4 \rangle$ repräsentiert. Sei J die Variablenzuweisung, sodass $x_1 = x_2 = \textit{wahr}$ und $x_3 = x_4 = \textit{falsch}$ gilt, also $J = (1, 1, 0, 0)$. Weiters ist N die aus der Vorlesung bekannte 1-flip Nachbarschaft für binäre Vektoren, also $J' \in N(J)$ falls sich J' von J in genau einem Bit unterscheidet.

(i) Bestimmen Sie $N(J)$ und geben Sie für jedes Element $J' \in N(J)$ an, wie viele Klauseln in G von J' erfüllt werden.

(ii) Ist J ein lokales Optimum für N? Begründen Sie Ihre Antwort kurz.

(iii) Kann eine lokale Suche angewandt auf G und J mit Nachbarschaftsstruktur N ein globales Optimum finden? Begründen Sie Ihre Antwort kurz.