TU Wien
Institut für Logic and Computation
Algorithms and Complexity Group

186.866 Algorithmen und Datenstrukturen VU 8.0

2. Test, 2022S

30. Juni 2022

Gruppe A

Nachfolgende Angaben gut **leserlich in BLOCKSCHRIFT** machen.

Nachname: ____________ Vorname: ____________

Matrikelnummer: ____________ Unterschrift: ____________

Sie dürfen die Lösungen nur auf diesen Prüfungsbogen schreiben. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie dokumentenechte Stifte (keine Bleistifte, etc.).

Die Verwendung von Taschenrechnern, Mobiltelefonen, Smartwatches, Tablets, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Kennzeichnen Sie bei Ankreuzfragen eindeutig, welche Kästchen Sie kreuzen. Streichen Sie Passagen, die nicht gewertet werden sollen, deutlich durch. Schreiben Sie leserlich, unleserliche Antworten werden nicht gewertet.

	A1	A2	A3	A4	A5	Summe
Erreichbare Punkte:	20	22	20	20	18	100
Erreichte Punkte:						

Viel Erfolg!

Aufgabe A1: P und NP, Spezialfälle **(20 Punkte)**

a) (8 Punkte) Betrachten Sie einen Graphen $G = (V, E)$. Eine Menge von Knoten $D \subseteq V$ heißt dominierende Menge, wenn jeder Knoten außerhalb von D adjazent zu mindestens einem Knoten in D ist. Das heißt, für jeden Knoten $u \in V \setminus D$ gibt es einen Knoten $v \in D$, sodass $(u, v) \in E$. Das DOMINATING SET Problem ist wie folgt definiert.

   DOMINATING SET: Gegeben sei ein Graph $G = (V, E)$ mit $|V| = n$ Knoten und eine ganze Zahl $k > 0$. Existiert eine dominierende Menge von Knoten $D \subseteq V$, sodass $|D| \leq k$?

   Wir wollen zeigen, dass DOMINATING SET in NP liegt, indem wir ein geeignetes Zertifikat und einen passenden Zertifizierer finden.

   (i) Welche Eigenschaften muss ein geeigneter Zertifizierer erfüllen?

   (ii) Geben Sie ein geeignetes Zertifikat und einen passenden Zertifizierer an. Argumentieren Sie, dass Ihr Zertifizierer die Eigenschaften aus Punkt (i) erfüllt.

b) (12 Punkte) Seien A, B und C Ja/Nein-Probleme und n deren Eingabegröße. Nehmen Sie an, es gibt

- eine Reduktion von 3-SAT nach DOMINATING SET in Zeit n^3,
- eine Reduktion von DOMINATING SET nach A in Zeit n,
- eine Reduktion von DOMINATING SET nach B in Zeit n^3,
- eine Reduktion von DOMINATING SET nach C in Zeit 4^n,
- eine Reduktion von A nach 3-SAT in Zeit n,
- eine Reduktion von C nach A in Zeit n^2.

Aus Unteraufgabe a) wissen wir, dass DOMINATING SET in NP liegt.

(i) Geben Sie die engste obere Schranke für die Laufzeit einer Reduktion von C auf DOMINATING SET in O-Notation an, die sich aus diesen Annahmen ableiten lässt. Begründen Sie Ihre Antwort kurz.

(ii) Was folgt aus den obigen Reduktionen für die Komplexität der Probleme DOMINATING SET, A, B, und C? Beantworten Sie die Frage, indem Sie **alle** zutreffenden Felder in der folgenden Tabelle ankreuzen.

	in P	in NP	NP-schwer	NP-vollständig
DOMINATING SET				
A				
B				
C				

Aufgabe A2: Branch-and-Bound **(22 Punkte)**

a) (4 Punkte) Welche Aussagen für Branch-and-Bound Verfahren sind korrekt? Kreuzen Sie Zutreffendes an.

(+1 Punkt für jede richtige, -1 Punkt für jede falsche und 0 Punkte für keine Antwort, keine negativen Gesamtpunkte auf diese Unteraufgabe)

(Q1) Eine korrekte optimale Lösung kann nur gefunden werden, wenn alle Teilprobleme betrachtet werden.

☐ Wahr ☐ Falsch

(Q2) Branch-and-Bound für ein Maximierungsproblem funktioniert im Allgemeinen umso besser, je größer die lokalen oberen Schranken sind.

☐ Wahr ☐ Falsch

(Q3) Wenn ein Branch-and-Bound Algorithmus die nächste Teilinstanz schlecht auswählt, kann es sein, dass keine optimale Lösung mehr gefunden wird.

☐ Wahr ☐ Falsch

(Q4) Bei einem Minimierungsproblem kann die Branch-and-Bound Suche für eine Teilinstanz abgebrochen werden, wenn die globale untere Schranke größer als lokale obere Schranke ist.

☐ Wahr ☐ Falsch

b) (8 Punkte) Wir betrachten einen Branch-and-Bound Algorithmus für ein Maximierungsproblem. Betrachten Sie die folgenden Teilinstanzen, die in diesem Branch-and-Bound Algorithmus behandelt werden könnten. Gehen Sie jeweils von einer globalen unteren Schranke von 70 aus, bevor die Schranken der Teilinstanz berechnet werden.

(I_1) Teilinstanz mit $L' = 60$, $U' = 80$

(I_2) Teilinstanz mit $L' = 78$, $U' = 78$

(I_3) Teilinstanz mit $L' = 55$, $U' = 55$

(I_4) Teilinstanz mit $L' = 40$, $U' = 65$

(I_5) Teilinstanz mit $L' = 70$, $U' = 75$

Geben Sie an, welche der Teilinstanzen im Branch-and-Bound Algorithmus weiter aufgespalten werden müssen.

Teilinstanzen: ☐

Bei welcher der Teilinstanzen würde sich die globale untere Schranke ändern?

Teilinstanzen: ☐

c) (8 Punkte) Betrachten Sie die folgende Instanz des Rucksackproblems. Die Gewichte und Werte der Gegenstände sind in der Tabelle angegeben. Die Kapazität des Rucksacks beträgt $G = 10$.

Gegenstand	A	B	C	D
Gewicht	3	7	2	6
Wert	15	21	14	12
Verhältnis				

Berechnen Sie die Wert-Gewichts-Verhältnisse aller Gegenstände und tragen Sie diese in obiger Tabelle ein. Geben Sie die Reihenfolge an, in der die Gegenstände betrachtet werden, wenn Sie den **verbesserten** Branch-and-Bound Algorithmus der Vorlesung anwenden.

Reihenfolge:

Betrachten Sie nun das initiale Teilproblem des **verbesserten** Branch-and-Bound Algorithmus aus der Vorlesung bei einer Rucksackkapazität von 10 und geben Sie die lokale obere Schranke U', die lokale untere Schranke L', sowie eine passende Auswahl von Gegenständen mit Gesamtwert L' an.

$L' =$

$U' =$

$S =$

d) (2 Punkte) Wie viele Teilinstanzen betrachtet der Branch-and-Bound Algorithmus für das Rucksackproblem im Allgemeinen im Worst-Case? Kreuzen Sie **alle** zutreffenden Schranken an, wobei n der Anzahl der Gegenstände entspricht.
(alles korrekt: 2 Punkte, sonst: 0 Punkte)

☐ $O(n^3)$ ☐ $O(n^2)$ ☐ $O(2^n)$ ☐ $\Theta(3^n)$

Aufgabe A3: Dynamisches Programmieren **(20 Punkte)**

Folgendes Problem ist als das GOLDGRÄBER Problem bekannt: Gegeben sei eine Matrix M der Dimension $m \times n$ die eine Goldmine repräsentiert. Jede Zelle der Matrix beinhaltet einen ganzzahligen Wert ≥ 0 der die Menge an Gold in dieser Zelle repräsentiert. Anfänglich wählt der Goldgräber eine beliebige Position in der ersten Spalte der Matrix. Der Goldgräber kann ab nun je einen von 3 Schritten durchführen:

- er bewegt sich diagonal nach rechts oben,
- er bewegt sich nach rechts,
- er bewegt sich diagonal nach rechts unten.

Schritte nach außerhalb der Matrix sind nicht erlaubt. Der Goldgräber stoppt sobald er in der letzten Spalte angekommen ist. Was ist die maximale Menge an Gold die der Goldgräber auf seinem Pfad einsammeln kann?

Für diese Aufgabe betrachten wir folgende Instanz:

$$M = \begin{Bmatrix} 1,3,1,5 \\ 2,2,4,1 \\ 5,0,2,2 \\ 0,6,1,0 \end{Bmatrix}$$

a) (5 Punkte) Nehmen Sie an, der Goldgräber wählt aus den erreichbaren Zellen immer eine mit dem meisten Gold.

(i) Um welche Art von Strategie handelt es sich hier?

(ii) Geben Sie die Zeilen in der Reihenfolge wie der Goldgräber sie besucht, sowie die eingesammelte Summe an Gold an. Der Index der Zeilen started bei 1.

(iii) Geben Sie eine 3x3 Matrix an bei der diese Strategie **immer** die optimale Lösung liefert.

b) (3 Punkte) Um einen optimalen Pfad des Goldgräbers Brute-Force zu finden, müssten alle Pfade durch die Matrix betrachtet werden. Geben Sie eine möglichst enge obere Schranke für die Gesamtanzahl der Pfade in Abhängigkeit von m Zeilen und n Spalten an. Sie können den Umstand ignorieren, dass Pfade außerhalb der Matrix ungültig sind.

c) (12 Punkte) Vervollständigen Sie folgendes dynamisches Programm welches eine optimale Lösung mithilfe einer Tabelle „gold“ berechnet. Wir nutzen folgende Variablen und Konstanten:

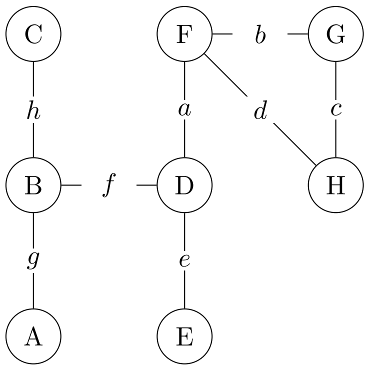
- Ein zweidimensionales Array M, das die Goldmine als Eingabe enthält.
- Die Anzahl der Zeilen m und die Anzahl der Spalten n.
- Die Variable i, welche den Index der aktuellen Zeile enthält.
- Die Variable j, welche den Index der aktuellen Spalte enthält.
- Ein zweidimensionales Array *gold* das im Eintrag (i, j) den maximal akkumulierten Wert an Gold speichert, der in einer Zeile i nach j-Schritten erreicht werden kann.
- Die Variablen *rightUp*, *right* und *rightDown* speichern für den aktuellen Wert von i und j das bis dahin akkumulierte Gold mit dem die Zelle (i, j) mit einem „diagonal nach rechts oben“, „nach rechts“ oder „diagonal nach rechts unten“ Zug erreicht werden kann.

Function `goldgräber(`M`)`
    Setze alle Einträge von *gold* auf 0
    for $j = 1 \ldots n$
        for $i = 1 \ldots m$
            if $j ==$ $\boxed{}$ **then**
                gold[i][j] $\leftarrow$ $\boxed{}$
            else
                right $\leftarrow$ $\boxed{}$
                if $i ==$ $\boxed{}$ **then**
                    rightUp $\leftarrow -\infty$ `//invalid move`
                else
                    rightUp $\leftarrow$ $\boxed{}$
                if $i ==$ $\boxed{}$ **then**
                    rightDown $\leftarrow -\infty$ `//invalid move`
                else
                    rightDown $\leftarrow$ $\boxed{}$
                gold[i][j] $\leftarrow$ $\boxed{}$

Aufgabe A4: Approximationsalgorithmen **(20 Punkte)**

a) Beim Minimum Vertex Cover Problem geht es darum, für einen gegebenen Graphen ein Vertex Cover mit minimaler Anzahl an Knoten zu finden. Betrachten Sie dazu den aus der Vorlesung bekannten 2-Approximationsalgorithmus.

(i) (6 Punkte) Führen Sie den Algorithmus auf folgendem Graphen aus und geben Sie das resultierende Vertex Cover an. Ist dieses minimal? Wenn Sie die Auswahl zwischen mehreren Kanten haben, nehmen Sie die alphabetisch kleinste.



(ii) (6 Punkte) Geben Sie zwei zusammenhängende Graphen G_1 und G_2 mit jeweils mindestens drei Knoten an, sodass **unabhängig von der Kantenwahl** im Algorithmus:

- für G_1 immer ein minimales Vertex Cover gefunden wird, also $c(G_1) = c_{\text{opt}}(G_1)$ gilt;
- für G_2 immer eine schlechtest mögliche Lösung mit $c(G_2) = 2 \cdot c_{\text{opt}}(G_2)$ gefunden wird.

b) (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 Gesamtpunkte auf diese Unteraufgabe)

(Q1) Falls P = NP gilt, dann existiert für das Minimum Vertex Cover Problem ein polynomieller 1.25-Approximationsalgorithmus.

☐ Wahr ☐ Falsch

(Q2) Da im symmetrischen TSP die Dreiecksungleichung gilt, gibt es dafür Approximationsalgorithmen mit besserer Gütegarantie als im metrischen TSP.

☐ Wahr ☐ Falsch

(Q3) Da das Finden eines Hamiltonkreises NP-vollständig ist, gibt es unter der Annahme von P $\neq$ NP keinen polynomiellen 2-Approximationsalgorithmus für das symmetrische TSP.

☐ Wahr ☐ Falsch

(Q4) Ein 2-Approximationsalgorithmus für ein Minimierungsproblem findet eine Lösung die höchstens doppelt so groß wie die optimale Lösung ist.

☐ Wahr ☐ Falsch

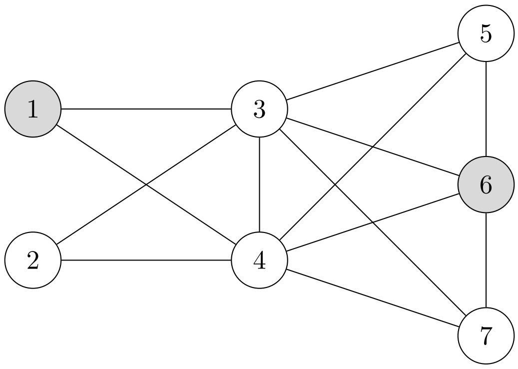
c) (4 Punkte) Für ein Maximierungsproblem $\mathcal{P}$ existieren zwei Approximationsalgorithmen A und B mit den Gütegarantien $\varepsilon_A = \frac{1}{2}$ und $\varepsilon_B = \frac{1}{3}$. Für eine Instanz I von $\mathcal{P}$ berechnen die Algorithmen die Werte $c_A(I) = 2022$ und $c_B(I) = 909$. In welchem Bereich kann der tatsächlich optimale Lösungswert $c_{\text{opt}}(I)$ liegen?

Aufgabe A5: Heuristische Verfahren **(18 Punkte)**

Beim Maximum Independent Set Problem geht es darum, für einen gegebenen Graphen $G = (V, E)$ ein Independent Set $S \subseteq V$ mit der maximalen Anzahl an Knoten zu finden. Wir definieren die folgenden zwei Nachbarschaftsstrukturen N_1 und N_2 für dieses Problem:

- $S' \in N_1(S)$ genau dann wenn S' aus S durch Hinzufügen eines einzigen Knotens in $V \setminus S$ erzeugt werden kann und noch immer ein Independent Set von G ist;
- $S' \in N_2(S)$ genau dann wenn $S' \in N_1(S)$ oder wenn S' durch Entfernen eines Knotens aus S und Hinzufügen von zwei Knoten aus $V \setminus S$ gebildet werden kann und noch immer ein Independent Set von G ist.

Gegeben ist folgender Graph H in dem $T = \{1, 6\}$ ein Independent Set ist.



a) (6 Punkte) Bestimmen Sie $N_1(T)$ und $N_2(T)$ für den obigen Graphen H.

b) (4 Punkte) Wenden Sie lokale Suche mit Nachbarschaftsstruktur N_2 auf H mit Ausgangslösung T an, bis ein lokales Optimum erreicht wird. Schrittfunktion und Durchmusterungsreihenfolge der Nachbarlösungen können frei gewählt werden. Geben Sie die Zwischenschritte an, also jedes Independet Set welches im Laufe der lokalen Suche ausgewählt wird.

c) (4 Punkte) Bestimmen Sie einen ungerichteten Graphen mit genau drei Knoten und eine Ausgangslösung (ein Independent Set), sodass eine lokale Suche mit N_1 kein globales Optimum erreicht, eine lokale Suche mit N_2 aber schon. Geben Sie eindeutig an, welche Knoten Ihres Graphen zur Ausgangslösung gehören.

d) (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 Gesamtpunkte auf diese Unteraufgabe)

(Q1) Lokale Suche mit N_1 ist ein $\frac{1}{2}$-Approximationsalgorithmus für das MAXIMUM INDEPENDENT SET Problem.

☐ Wahr ☐ Falsch

(Q2) Sei G ein beliebiger Graph und S ein Independent Set von G. Dann gilt $|N_1(S)| \leq |N_2(S)|$.

☐ Wahr ☐ Falsch

(Q3) Angenommen man findet bei einer lokalen Suche mit N_1 ein maximales Independent Set S für einen Graphen G. Dann ist S auch ein minimales Vertex Cover für G.

☐ Wahr ☐ Falsch

(Q4) Angenommen man könnte bei lokaler Suche mit N_2 immer garantiert ein globales Optimum (ein maximales Independent Set) finden, egal welcher Graph und welche Ausgangslösung verwendet werden. Dann gilt P = NP.

☐ Wahr ☐ Falsch