

2019_t2_A

Disclaimer

Alles was hier drinnen steht kann Fehler enthalten!, Falls dir etwas auffällt melde dich gerne auf Discord bei mir ([@xmozz](#))

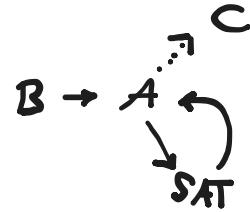
A1: P und NP

Stoff: 9. Polynomialzeitreduktionen

a)

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

- eine Reduktion von B nach A in Zeit $O(n^2)$,
- eine Reduktion von A nach SAT in Zeit $O(n^3)$,
- eine Reduktion von SAT nach A in Zeit $O(n)$,
- eine Reduktion von A nach C in Zeit $O(3^{2n})$.



- (i) Kreuzen Sie in den folgenden Tabellen jeweils die zutreffenden Felder an:
(je korrekter Zeile 1 Punkt, keine Minuspunkte)

C ist ...	Ja	Nein	Keine Aussage möglich
... in NP			X
... NP-schwer			X

A ist ...	Ja	Nein	Keine Aussage möglich
... in NP	X		
... NP-schwer	X		

B ist ...	Ja	Nein	Keine Aussage möglich
... in NP	X		
... NP-schwer			X

- (ii) Welche der obigen Probleme A, B, C und SAT wären sicher in polynomieller Zeit lösbar, falls $P = NP$ gelten sollte.

A, B, SAT

- (iii) Nehmen Sie nun an, C kann für Eingaben der Größe m in Zeit $O(m^2)$ gelöst werden. Welche engste obere Schranke können Sie aus den gegebenen Informationen für die Worst-Case Laufzeit eines optimalen Algorithmus für A schließen?

$$\mathcal{O}((3^{2n})^2) = \underline{\mathcal{O}(3^{4n})}$$

b)

- b) (9 Punkte) Im Folgenden sind verschiedene Probleme mit verschiedenen Zertifikaten gegeben. Welche Zertifikate sind (gemeinsam mit einem geeigneten Zertifizierer) geeignet, um zu zeigen, dass das gegebene Problem in NP ist? Kreuzen Sie Zutreffendes an. Sie dürfen für diese Frage P \neq NP annehmen.

Es kann mehr als eine richtige Antwort geben.

(je Unteraufgabe: alles korrekt: 3 Punkte, ein Fehler: 1 Punkt, sonst / kein Kreuz: 0 Punkte)

- i) **Problem:** Gegeben eine Menge U und eine Menge $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$ von Teilmengen von U . Gibt es ein Set Cover von U mit $\leq k$ Mengen?

Zertifikat:

- Ein leerer String.
- Ein Set Cover mit $\leq k$ Mengen.
- Eine Zahl m , so dass $m \leq k$ und es ein Set Cover mit m Mengen gibt.
- Keines der Zertifikate ist geeignet.

- ii) **Problem:** Gegeben sei ein Graph G mit gewichteten Kanten. Gibt es einen Spanning Tree von G mit Gewicht $\leq k$?

Zertifikat:

- Ein leerer String.
- Ein Spanning Tree von G mit Gewicht $\leq k$.
- Alle möglichen Teilmengen von Kanten. *wäre zu groß*
- Keines der Zertifikate ist geeignet.

- iii) **Problem:** Gegeben sei ein Graph G . Hat das *größte* Independent Set von G genau k Knoten?

Zertifikat:

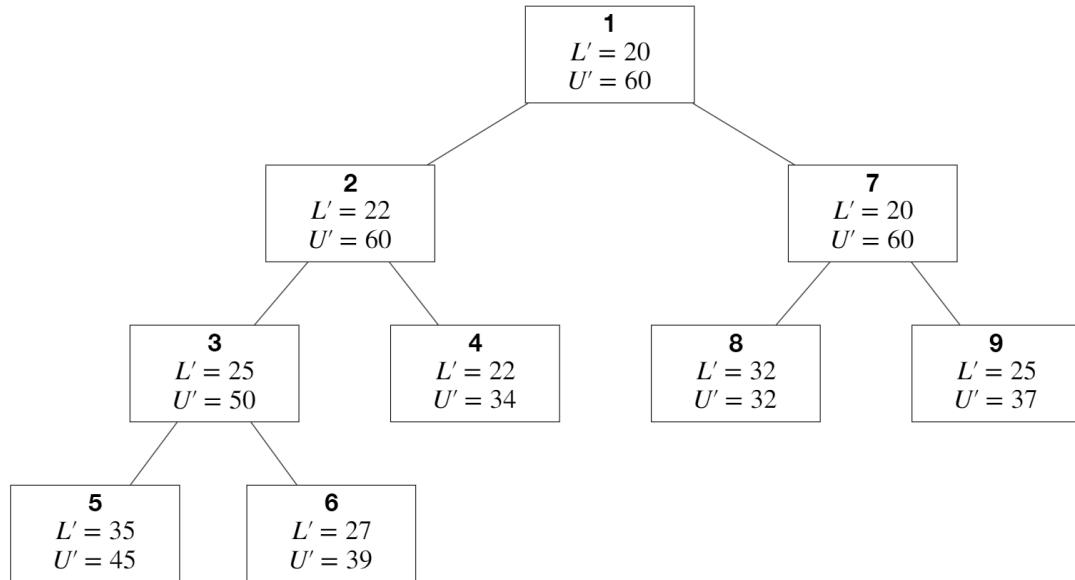
- Die Größe l des größten Independent Sets.
- Ein Independent Set mit $> k$ Knoten.
- Ein Independent Set mit k Knoten.
- Keines der Zertifikate ist geeignet.

A2: Branch-and-Bound

Stoff: sem_2/AlgoDat/vo/2. Test/11. Branch and Bound

a)

- a) (7 Punkte) Branch-and-Bound wird zur Lösung eines *Minimierungsproblems* verwendet und erzeugt den unten abgebildeten Suchbaum. Jeder Knoten entspricht einer Teilinstanz mit einer lokalen unteren Schranke L' und einer lokalen oberen Schranke U' .



Geben Sie die Menge S der Nummern jener Teilinstanzen (repräsentiert durch Blattknoten) an, die *nicht* mehr weiter aufgespalten werden müssen.

S: **8, 5** weil 5 Untere Schranke größer als globale Obere und 8 ist Lösung.

Was ist die globale obere Schranke U ?

U: **32**

b)

- b) (2 Punkte) Was ist die Idee / das Prinzip der Lokalen Suche?

Ausnahmsweise auch kurzfristig schlechtere Teillösungen akzeptieren um zu einem lokalen Optimum zu kommen.

Das macht man ausgehend von einer momentanen Lösung und man schaut sich dann die Nachbarlösungen von der jetzigen Lösung an.

c)

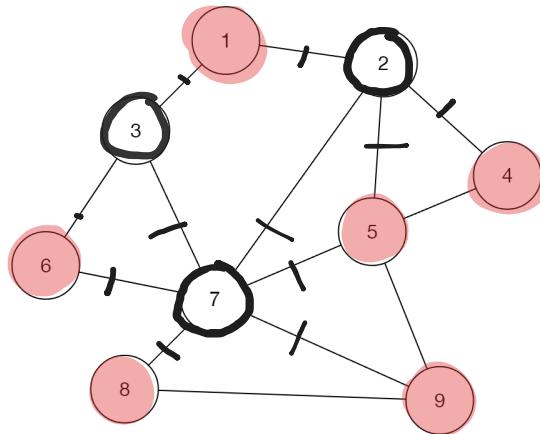
- c) (2 Punkte) Nennen Sie einen Unterschied zwischen Heuristischen Verfahren und Approximationsalgorithmen, den Sie aus der Vorlesung kennen.

Mit Hilfe von Heuristischer Verfahren versucht man an eine optimale Lösung zu kommen und Approximationsalgorithmen geben eine Lösung mit einer gewissen Gütegarantie. Also man bekommt eine Lösung die Optimal sein kann, aber bis zu einem gewissen Faktor schlechter sein kann als das Optimum.

Der Hauptunterschied liegt in der Gütegarantie: **Heuristische Verfahren** versuchen, eine *gute* Lösung in endlicher Zeit zu finden, bieten aber **keine Garantie** dafür, wie nah diese Lösung am Optimum ist oder ob sie überhaupt optimal ist. **Approximationsalgorithmen** hingegen liefern eine Lösung mit einer **Gütegarantie**, d.h. sie garantieren, dass die gefundene Lösung höchstens um einen bestimmten Faktor schlechter ist als die optimale Lösung

d)

- d) (7 Punkte) Auf dem folgenden Graphen wird eine obere und eine untere Schranke für die Größe eines kleinsten Vertex Covers berechnet.

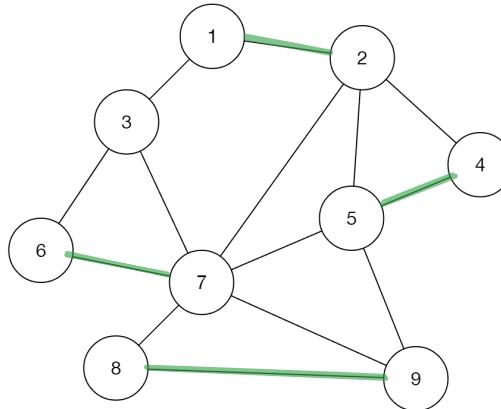


- (i) Wenden Sie die Greedy-Heuristik, die jeweils einen Knoten mit höchstem Grad wählt, an und geben Sie die dabei ausgewählte Menge an Knoten an.

Ausgewählte Knoten:

7, 2, 3

- (ii) Berechnen Sie ein nicht erweiterbares Matching und zeichnen Sie das Matching im Graphen ein.



- (iii) Für die Größe eines kleinsten Vertex Covers ist das eben berechnete nicht erweiterbare Matching ...

... eine obere Schranke.

... eine untere Schranke.

(korrekt: 1 Punkte, inkorrekt: -1 Punkt, nicht beantwortet: 0 Punkte)

e)

e) (3 Punkte) In welchen Fällen kann die Branch-and-Bound Suche für eine Teilinstanz bei einem Maximierungsproblem abbrechen? Kreuzen Sie Zutreffendes an.

(alles korrekt: 3 Punkte, ein Fehler: 1 Punkt, sonst / kein Kreuz: 0 Punkte)

- Globale untere Schranke ist größer als lokale obere Schranke.
- Lokale untere Schranke ist größer als globale untere Schranke.
- Lokale obere Schranke entspricht lokaler unterer Schranke.
- Lokale untere Schranke ist kleiner als globale untere Schranke.

A3: NP-Vollständigkeit Spezialfälle

Stoff: 10. NP-Vollständigkeit Spezialfälle

a)

- a) (12 Punkte) *Gewichtetes Vertex Cover auf Bäumen* kann, analog zu gewichtetem Independent Set auf Bäumen, mittels dynamischer Programmierung gelöst werden.

Kreuzen Sie im nachfolgenden Pseudocode jene Codezeilen an, die ausgeführt werden müssen, um eine funktionierende Implementierung von folgendem Algorithmus für gewichtetes Vertex Cover auf Bäumen zu erhalten: der Algorithmus erhält einen Baum $T = (V, E)$, bei dem jedem Knoten $v \in V$ ein positives Gewicht w_v zugewiesen ist und berechnet das minimale Gewicht eines Vertex Covers für T .

Je Block (grau hinterlegte Box) ist genau eine Auswahl korrekt.

(je Block: korrekt: 3 Punkte, falsch / mehrere Kreuze: -1 Punkt, kein Kreuz 0 Punkte. Zumindest 0 Punkte für diese Unteraufgabe)

Min-Weight-Vertex-Cover-In-A-Tree(T):

Wähle eine Wurzel r aus

foreach Knoten u von T **do**

$A_{\text{out}} \leftarrow \infty; A_{\text{in}} \leftarrow \infty$

foreach Knoten u von T in Postorder **do**

foreach Knoten u von T in Preorder **do**

foreach Knoten u von T in Inorder **do**

if u ist ein Blatt **then**

$\begin{cases} A_{\text{in}}[u] \leftarrow 0 \\ A_{\text{out}}[u] \leftarrow w_u \end{cases}$

$\begin{cases} A_{\text{in}}[u] \leftarrow 1 \\ A_{\text{out}}[u] \leftarrow 0 \end{cases}$

$\begin{cases} A_{\text{in}}[u] \leftarrow w_u \\ A_{\text{out}}[u] \leftarrow 0 \end{cases}$

else

$\begin{cases} A_{\text{in}}[u] \leftarrow 1 + \sum_{v \in \text{Nachfolger}(u)} A_{\text{in}}[v] \\ A_{\text{out}}[u] \leftarrow w_u + \sum_{v \in \text{Nachfolger}(u)} \min\{A_{\text{in}}[v], A_{\text{out}}[v]\} \end{cases}$

$\begin{cases} A_{\text{in}}[u] \leftarrow w_u + \sum_{v \in \text{Nachfolger}(u)} \min\{A_{\text{in}}[v], A_{\text{out}}[v]\} \\ A_{\text{out}}[u] \leftarrow \sum_{v \in \text{Nachfolger}(u)} A_{\text{in}}[v] \end{cases}$

$\begin{cases} A_{\text{in}}[u] \leftarrow w_u + \sum_{v \in \text{Nachfolger}(u)} \max\{A_{\text{in}}[v], A_{\text{out}}[v]\} \\ A_{\text{out}}[u] \leftarrow \sum_{v \in \text{Nachfolger}(u)} A_{\text{in}}[v] \end{cases}$

return $\min\{A_{\text{in}}[r], A_{\text{out}}[r]\}$

return $A_{\text{in}}[r] + A_{\text{out}}[r]$

return $\max\{A_{\text{in}}[r], A_{\text{out}}[r]\}$

b)

- b) (8 Punkte) Nehmen Sie an, dass gewichtetes Vertex Cover auf Bäumen in quadratischer Zeit gelöst werden kann.

Kreuzen Sie an, ob die folgenden (voneinander unabhängigen) Aussagen wahr oder falsch sind.

(je Unteraufgabe: korrekt: 2 Punkte, falsch: -2 Punkte, kein Kreuz: 0 Punkte. Zu mindest 0 Punkte für diese Unteraufgabe)

- (i) Daraus folgt, dass gewichtetes Vertex Cover auf *Graphen* in quadratischer Zeit gelöst werden kann.

Wahr Falsch

- (ii) Daraus folgt, dass *ungewichtetes* Vertex Cover auf Bäumen in quadratischer Zeit gelöst werden kann.

Wahr Falsch

- (iii) Daraus folgt, dass gewichtetes Vertex Cover auf *Pfaden* in quadratischer Zeit gelöst werden kann.

Wahr Falsch

- (iv) Daraus folgt, dass gewichtetes Vertex Cover auf *Binärbäumen* in quadratischer Zeit gelöst werden kann.

Wahr Falsch

A4: Dynamisches Programmieren

Stoff: sem_2/AlgoDat/vo/2. Test/12. Dynamische Programmierung

a)

- a) (9 Punkte) Betrachten Sie das aus der Vorlesung bekannte Rucksackproblem. Gegeben sei eine Instanz mit einer Kapazität $G = 5$ und den folgenden fünf Gegenständen, die jeweils nur einmal vorhanden sind:

	Gegenstand				
	A	B	C	D	E
Wert	1	3	2	4	6
Gewicht	2	1	1	3	2

Vervollständigen Sie die untenstehende Tabelle, sodass der Eintrag in Zeile i und Spalte g gerade dem Wert $OPT(i, g)$ entspricht, der sich mit den ersten i Gegenständen und einem Rucksack der Kapazität g erreichen lässt.

		Kapazität					
		0	1	2	3	4	5
		0	0	0	0	0	0
\emptyset		0	0	0	0	0	0
{A}		0	0	1	1	1	1
{A, B}		0	3	3	4	4	4
{A, B, C}		0	4	7	7	9	9
{A, B, C, D}		0	4	7	7	9	11
{A, B, C, D, E}		0	4	7	10	12	12

Was ist der optimale Wert einer Lösung für einen Rucksack mit Kapazität $G = 3$?

Wert:

10

Angenommen der Gegenstand E ist nicht mehr verfügbar. Welche Gegenstände entsprechen nun einer optimalen Lösung bei Kapazität $G = 5$?

Gegenstände:

B, C, D

b)

- b) (9 Punkte) Sei nun eine weitere Instanz des Rucksackproblems gegeben mit Kapazität $G = 6$ und den folgenden sechs Gegenständen.

	Gegenstand					
	A	B	C	D	E	F
Wert	2	5	3	1	6	9
Gewicht	2	1	1	2	4	3

Die Wertetabelle M nach Ausführung des Dynamischen Programms lautet wie folgt:

Gegenstände	Kapazität							
	0	1	2	3	4	5	6	
\emptyset	0	0	0	0	0	0	0	0
$\{A\}$	0	0	2	2	2	2	2	2
$\{A, B\}$	0	5	5	7	7	7	7	7
$\{A, B, C\}$	0	5	8	8	10	10	10	10
$\{A, B, C, D\}$	0	5	8	8	10	10	11	
$\{A, B, C, D, E\}$	0	5	8	8	10	11	14	
$\{A, B, C, D, E, F\}$	0	5	8	9	14	17	17	

- (i) Markieren Sie in der Tabelle all jene Felder, die der Algorithmus **Find-Solution(M)** aus der Vorlesung bei der Berechnung der Lösungsmenge S ausliest und verwendet.
- (ii) Geben Sie die Menge der Gegenstände in der Lösungsmenge S an.

$$S = \{F, C, B\}$$

c)

- c) (7 Punkte) Gegeben ist ein gerichteter Graph $G = (V, E)$ mit $|V| = n$ Knoten. Die Kanten von G sind mit Kantengewichten $c_{vw} \in \mathbb{R}$ für alle Kanten $(v, w) \in E$ gewichtet, sodass es keine negativen Kreise gibt.

- (i) Ergänzen Sie Bellmans Rekursionsgleichungen für die Länge $OPT(i, v)$ eines kürzesten $v-t$ Pfades in G mit höchstens i Kanten, wobei $v \in V$ ein beliebiger Knoten ist und $t \in V$ ein fester Zielknoten.

$$OPT(0, t) =$$


$$OPT(0, v) =$$

für $v \neq t$

$$OPT(i, v) = \min_{(v,w) \in E} (c_{vw} + OPT(i-1, w))$$
für $i \geq 1$

- (ii) Geben Sie den kleinsten Wert für die Kantenanzahl i an, sodass für jeden Graphen G , der die gegebenen Voraussetzungen erfüllt, Folgendes gilt:

$$OPT(i, v) = OPT(i+1, v) \text{ für alle Knoten } v \in V.$$

Der kleinste Wert ist $i =$ **$n - 1$** .

- (iii) Geben Sie für Ihre Wahl der kleinsten Kantenanzahl i (aus Unterpunkt (ii)) eine *kurze* Begründung in 1–2 Sätzen an.

In einem gerichteten Graphen ohne negative Zyklen enthält ein kürzester Pfad höchstens $n-1$ Kanten. Nach $n-1$ Iterationen des Bellman-Ford-Algorithmus sind die Längen aller kürzesten Pfade gefunden, da sich die Pfadlängen danach nicht mehr verbessern können.

A5: Approximationsalgorithmen

Stoff: 13. Approximation

a)

- a) (6 Punkte) Angenommen A ist ein ε -Approximationsalgorithmus für ein Minimierungsproblem (d.h. $\varepsilon \geq 1$). Sei x eine Probleminstanz mit optimalem Lösungswert $C_{OPT}(x)$ und $C_A(x)$ der Wert der von A berechneten Lösung. Welche der folgenden Aussagen treffen in jedem Fall zu? Kreuzen Sie Zutreffendes an.

- $C_{OPT}(x) \leq C_A(x)$
- $C_A(x) > C_{OPT}(x)$
- $C_A(x) \leq \varepsilon \cdot C_{OPT}(x)$
- $C_A \leq \frac{C_{OPT}(x)}{\varepsilon}$

Sei B ein Approximationsalgorithmus für das gleiche Problem mit Gütegarantie $\varepsilon/2$ und $C_B(x)$ der von B berechnete Lösungswert für die Instanz x . Welche der folgenden Aussagen sind in jedem Fall korrekt? Kreuzen Sie Zutreffendes an.

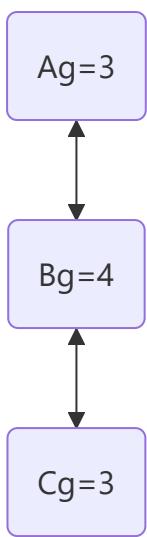
- $\frac{C_A(x)}{C_B(x)} \geq 2$
- $\frac{C_B(x)}{C_A(x)} = \frac{1}{2}$
- $C_B(x) \leq \varepsilon \cdot C_{OPT}(x)$
- $C_A(x) + C_B(x) \leq 3\varepsilon/2 \cdot C_{OPT}(x)$

(je Block: alles korrekt: 3 Punkte, ein Fehler: 1 Punkte, sonst / kein Kreuz: 0 Punkte)

b)

- b) (7 Punkte) Beim Gewichteten Independent Set Problem wird in einem Eingabegraben nach einem Independent Set mit maximalem Gewicht gesucht (jedem Knoten ist ein Gewicht zugeordnet, das Gewicht eines Independent Set ist die Summe der Gewichte der darin enthaltenen Knoten). Zeigen Sie, dass ein Greedy-Algorithmus für Gewichtetes Independent Set, der wiederholt einen Knoten mit maximalem Gewicht zu einem bestehenden Independent Set hinzufügt, jede konstante Gütegarantie $\varepsilon \leq 1$ verletzt.

Nehmen wir einen Graphen G an:



Hier würde der Greedy Algorithmus den Knoten B nehmen und die anderen beiden ein größeres Gewicht zusammen haben und auch noch ein größeres Independent set sind, werden dann nicht genommen.

Außerdem, wenn der GA nicht mehr macht als einfach nur die Knoten mit höchsten Gewicht hinzuzufügen, dann wird das Ergebnis ziemlich wahrscheinlich auch kein IS sein, da der Algorithmus dann nicht stoppen würde bis jeder Knoten im IS drinnen ist, was dann kein IS ist, wenn die Knoten Kanten zwischen sich haben.

Gegenbeispiel:

Gegeben sei ein Stern-Graph: Zentrum z , Blätter v_1, \dots, v_n mit Gewichten $w(z) = n$, $w(v_i) = n - 1$.

- Greedy nimmt z : Gewicht n
- Optimales IS: $\{v_1, \dots, v_n\}$, Gewicht $n(n - 1)$

Dann gilt:

$$\frac{w(\text{Greedy})}{w(\text{OPT})} = \frac{n}{n(n - 1)} = \frac{1}{n - 1} \rightarrow 0$$

Also verletzt der Greedy-Algorithmus jede konstante Gütegarantie $\varepsilon \leq 1$.