



Komplexitätstheorie WS 00/01 Lösungsvorschläge zur Klausur

Aufgabe 1

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
JNN	NJJ	JNJ	NNJ	JNN	JNJ	JNJNNJ	JJJ	JJJ

Aufgabe 7

Wenn für eine Eingabe x die Anzahl der von WFD benötigten Behälter 1 ist, ist die Behauptung gezeigt. Nehmen wir im folgenden an, dass WFD mehr als einen Behälter benötigt.

Wir definieren zur Abkürzung $S := \sum_{i=1}^n a_i$.

Man macht zunächst folgende Beobachtungen:

1. Ganz egal, wie eine Verpackung mit r Behältnissen aussieht, die Behälter müssen immer mindestens so viel Platz bieten, wie Volumen zu verpacken ist. Also muss gelten

$$r \cdot b \geq S,$$

und da bei uns $b = 1$ und die letzte Gleichung auch für $r = \text{opt}(x)$ erfüllt ist, erhalten wir

$$(1) \quad \text{opt}(x) \geq S.$$

2. Sei nun $k := \text{WFD}(x)$ die Anzahl der Behälter, die WFD benutzt, um die Objekte der Eingabe x zu verpacken. Die Füllstände der Behälter seien b_1, \dots, b_k . Dann gilt für beliebige i, j mit $1 \leq i, j \leq k$ und $i \neq j$, dass $b_i + b_j > 1$, denn sonst hätte WFD den Inhalt des einen Behälters mit in den anderen verpackt.
3. Das zu verpackende Volumen S ist gleich der Summe der Füllstände der k Behälter, also gilt:

$$(2) \quad S = \sum_{j=1}^k b_j.$$

Die letzte Gleichung erweitern wir mit 2 und erhalten:

$$(3) \quad 2S = 2 \sum_{j=1}^k b_j = \underbrace{b_1 + b_2 + \dots + b_k}_{>1} + b_1 + \dots + \underbrace{b_{k-1} + b_k}_{>1}$$

Auf der rechten Seite dieser Ungleichung hat man k Paare stehen, deren Summe jeweils > 1 ist (Beobachtung 2), also ist die rechte Seite $> k$.

Schließlich erhalten wir daraus zusammen mit der ersten Beobachtung:

$$(4) \quad 2\text{opt}(x) \geq S > k = \text{WFD}(x)$$

und damit die gewünschte Behauptung. \square

Aufgabe 8

Die Idee für \mathcal{A}_2 ist folgende: Wir berechnen zunächst die kleinste Anzahl von Zeilen k^* , die man benötigt, um alle Spalten mit einer 1 zu überdecken. Dazu ruft man einmal \mathcal{A}_1 auf: $k^* := \mathcal{A}_1(k)$.

Betrachtet man nun eine Matrix B , die durch Streichen einer beliebigen Zeile i aus A hervorgeht, so gibt es folgende Fälle:

1. B hat in jeder Spalte eine 1. Dann berechne man $k' := \mathcal{A}_1(B)$. Nun gibt es wieder zwei Fälle:
 - $k' = k$. Dann gibt es eine Überdeckung mit der minimalen Anzahl von Zeilen, in der die Zeile i nicht benötigt wird.
 - $k' > k$. Dann wird die Zeile i in der Überdeckung mit der minimalen Anzahl von Zeilen benötigt.
2. B hat nicht in jeder Spalte eine 1. Dann wird die Zeile i in jeder Auswahl, die eine vollständige Überdeckung der Spalten bildet, benötigt.

Somit braucht man nur für jede Zeile der Matrix A testen, ob diese in einer minimalen Überdeckung vorkommen muss oder nicht. Wird eine solche Zeile nicht benötigt, entfernt man sie aus der Matrix A . Nachdem man alle Zeilen getestet hat, ist A die Matrix mit der kleinsten Anzahl von Zeilen, welche alle Spalten überdeckt.

Da nun nach den Zeilennummern der Zeilen in der minimalen Überdeckung gefragt war, muss man noch eine Kleinigkeit beachten: Wirft man eine Zeile

aus der Matrix heraus, verschieben sich die Zeilennummern um 1, und das macht Ärger. Deswegen beginnt man beim Untersuchen der Matrix mit der letzten Zeile. (Beim Korrigieren der Lösungen war uns dieses Detail aber nicht wichtig, hat also keine Punktabzüge verursacht).

Schließlich erhält man den Algorithmus 1:

Algorithmus 1 Algorithmus \mathcal{A}_2 für Aufgabe 8

Eingabe: Matrix $A \in \{0, 1\}^{n \times m}$, die in jeder Spalte eine 1 enthält.

Ausgabe: Auswahl $I \in \{1, \dots, n\}$ von Zeilen von A , die eine minimale Überdeckung bilden.

$I := \emptyset$

$k^* := \mathcal{A}_1(A)$

for $i := n, n - 1, \dots, 1$ **do**

$B := A$ ohne Zeile i

if B hat in jeder Spalte eine 1 **then**

if $\mathcal{A}_1(B) = k^*$ **then**

$A := B$ {Zeile i wird nicht gebraucht und aus A rausgeworfen}

else

$I := I \cup \{i\}$ {Zeile i wird gebraucht}

else

$I := I \cup \{i\}$ {Zeile i wird gebraucht}

return I
