

Lösungsvorschläge – Blatt 4

Zürich, 21. März 2024

Lösung zu Aufgabe 4

(a) Wir betrachten den folgenden Algorithmus P-MP für MINIMUM PARTITION:

Eingabe: $a_1, a_2, \dots, a_n \in \mathbb{N}$ mit $a_1 \geq a_2 \geq \dots \geq a_n$, $\varepsilon > 0$.

Schritt 1: $k := \left\lceil \frac{1}{\varepsilon} \right\rceil$

Schritt 2: Für jede Menge $S \subseteq \{1, \dots, k\}$:

Schritt 2.1: $I := S$;

$J := \{1, \dots, k\} - S$;

$SI := \sum_{i \in I} a_i$;

$SJ := \sum_{i \in J} a_i$;

Schritt 2.2:

for $i := k + 1$ **to** n

if $SI \leq SJ$

then $I := I \cup \{i\}$; $SI := SI + a_i$

else $J := J \cup \{i\}$; $SJ := SJ + a_i$

Schritt 2.3: Die jeweils bisher beste Lösung I wird gespeichert.

Ausgabe: Diejenige Menge I aus Schritt 2 mit den geringsten Kosten.

(b) Schritt 1 kann man in $\mathcal{O}(1)$ durchführen. Die Anzahl der Teilmengen S in Schritt 2 ist 2^k . Wenn man die Mengen in geeigneter Reihenfolge aufzählt, dann kann jede Menge in $\mathcal{O}(1)$ aus der vorangegangenen Menge konstruiert werden. Das Erweitern einer Menge S zu einer Lösung ist in $\mathcal{O}(n)$ möglich. Wegen $n \geq k$ folgt

$$\text{Time}_{\text{P-MP}}(n) \leq \mathcal{O}(2^k \cdot n) = \mathcal{O}\left(2^{\left\lceil \frac{1}{\varepsilon} \right\rceil} \cdot n\right).$$

(c) Sei $d_i = |\sum_{t \in \{1, \dots, i\} \cap I} a_t - \sum_{t \in \{1, \dots, i\} \setminus I} a_t|$ der Unterschied (dem Betrag nach) der beiden Teillösungen nach dem Verteilen der Zahlen $a_1, \dots, a_i$, für $i \geq k$. Insbesondere ist d_k der Wert vor dem Starten der Greedy-Strategie. Aufgrund der Greedy-Vorgehensweise gilt für $i > k$:

1. Falls $a_i \leq d_{i-1}$, so ist in der kleineren Summe Platz für a_i , ohne dadurch die bisher grössere zu übertreffen, also ist $d_i = d_{i-1} - a_i$.
 2. Sonst wird die kleinere Summe zur grösseren. Es folgt $a_i = d_{i-1} + d_i \geq d_i$.
- (d) Wir zeigen per Induktion über $t = j - i$, dass, falls $a_i \geq d_i$ für ein $i > k$ gilt, dann auch $a_i \geq d_j$ für alle $j \geq i$ gilt.

Für $t = 0$ ist nichts zu zeigen. Wir nehmen jetzt an, dass die Behauptung für alle $0 \leq t' \leq t$ gilt und zeigen sie für $t + 1$. Wir unterscheiden die zwei Fälle aus Aufgabenteil (c): Falls $a_{i+t+1} \geq d_{i+t+1}$, gilt auch $a_i \geq a_{i+t+1} \geq d_{i+t+1}$, weil die a_i absteigend sortiert sind. Andernfalls gilt $d_{i+t+1} = d_{i+t} - a_{i+t} < d_{i+t}$, und die Behauptung folgt direkt aus der Induktionsannahme.

- (e) Falls $n \leq k$, findet P-MP stets die optimale Lösung. Es sei im Folgenden also $n > k$. Sei Opt die optimale Lösung. $S_O = \text{Opt} \cap \{1, \dots, k\}$ wird als Startmenge von P-MP verwendet und per Greedy-Strategie zu I_O aufgefüllt. Wir betrachten im Folgenden diese Lösung I_O .

Falls $\sum_{j=k+1}^n a_j \leq d_k$, so werden alle Zahlen $a_{k+1}, \dots, a_n$ von der Greedy Strategie zu der gleichen Summe hinzugefügt, und dies ist offenbar auch die optimale Lösung, d.h. $I_O = \text{Opt}$. Andernfalls tritt Fall 2 aus Aufgabenteil (c) für ein $i \in \{k+1, \dots, n\}$ auf und es folgt

$$d_n \leq a_i \leq a_{k+1}. \quad (1)$$

Ausserdem folgt aus $a_1 \geq a_2 \geq \dots \geq a_{k+1}$, dass

$$a_{k+1} \leq \frac{1}{k+1} \sum_{j=1}^n a_j. \quad (2)$$

Die Kosten der berechneten Lösung I_O lassen sich berechnen als

$$\text{cost}(I_O) = \frac{1}{2} \cdot \left(d_n + \sum_{i=1}^n a_i \right).$$

Damit erhalten wir

$$\begin{aligned} \text{cost}(I_O) &= \frac{\sum_{j=1}^n a_j + d_n}{2} \stackrel{(1)}{\leq} \frac{\sum_{j=1}^n a_j + a_{k+1}}{2} \stackrel{(2)}{\leq} \left(1 + \frac{1}{k+1} \right) \frac{\sum_{j=1}^n a_j}{2} \\ &\leq \left(1 + \frac{1}{k+1} \right) \text{cost}(\text{Opt}), \end{aligned}$$

da die optimale Lösung mindestens die Hälfte der Gesamtsumme kostet. Weil I_O eine der von P-MP konstruierten Lösungen ist, gilt insgesamt

$$R_{\text{P-MP}} \leq \frac{\left(1 + \frac{1}{k+1} \right) \text{cost}(\text{Opt})}{\text{cost}(\text{Opt})} = 1 + \frac{1}{k+1} \leq 1 + \varepsilon.$$

Lösung zu Aufgabe 5

Die Idee unseres Greedy-Algorithmus App-MSKP ist, den Rucksack mit vielen Exemplaren der kleinsten Sorte zu füllen, ausser wenn nur ein Objekt in den Rucksack passt.

Eingabe: Positive ganze Zahlen $w_1, \dots, w_n, b$, mit $w_1 \leq b, \dots, w_n \leq b$.

1. Sortiere alle w_i absteigend (oBdA sei danach $b \geq w_1 \geq \dots \geq w_n$).
2. **if** $w_n > b/2$
 $a_1 := 1, a_2 := \dots := a_n := 0$.
 else
 $a_1 := \dots := a_{n-1} := 0$ und $a_n := \lfloor b/w_n \rfloor$.

Ausgabe: $(a_1, \dots, a_n)$.

Nun zeigen wir, dass App-MSKP ein polynomieller 1.5-Approximationsalgorithmus ist.

Die Laufzeit ist durch das Sortieren im ersten Schritt beschränkt, da im zweiten Schritt nur linear viele Wertzuweisungen stattfinden. Die Laufzeit ist also in $O(n \log n)$. Wenn man statt zu sortieren nur das grösste und das kleinste Objekt sucht (die anderen werden nicht benötigt), geht das sogar in Linearzeit.

Die Approximationsgüte kann folgendermassen bestimmt werden: Falls $w_n > b/2$, sind alle Objekte grösser als $b/2$, also kann nur eines einmal eingepackt werden, und es ist optimal, das grösste einmal zu nehmen. Das tut der Algorithmus in diesem Fall.

Falls $b/2 \geq w_n > b/3$, so ist $a_n := \lfloor b/w_n \rfloor = 2$ und somit $a_n w_n > 2b/3$. Der Wert der optimalen Lösung Opt ist nach oben beschränkt durch die Kapazität b des Rucksacks, also ist die Güte der berechneten Lösung in diesem Fall

$$\frac{Opt}{cost(a_1, \dots, a_n)} \leq \frac{b}{a_n w_n} < \frac{b}{2b/3} = \frac{3}{2}.$$

Falls schliesslich $w_n \leq b/3$, so bleibt im Rucksack weniger als w_n Platz frei, denn sonst hätten wir das Objekt der Grösse w_n noch ein weiteres Mal eingepackt. Also ergibt sich auch in diesem Fall eine Approximationsgüte von

$$\frac{Opt}{cost(a_1, \dots, a_n)} < \frac{b}{b - w_n} \leq \frac{b}{b - b/3} \leq \frac{b}{2b/3} = \frac{3}{2}.$$

Somit ist App-MSKP ein 3/2-Approximationsalgorithmus.