

Lösungsstrategien für NP-schwere Probleme der Kombinatorischen Optimierung

— Übungsblatt 6 —

Walter Stieben
(4stieben@inf)

Tim Reipschläger
(4reipsch@inf)

Louis Kobras
(4kobras@inf)

Hauke Stieler
(4stieler@inf)

Abgabe am: 30. Mai 2016

Aufgabe 6.1

a)

Zu zeigen ist, dass der angegebene Algorithmus kein 2-Approximationsalgorithmus ist. Zeigen kann man das mit einem Gegenbeispiel:

Sei $A = \{1, 2, 8\}$ und $B = 10$. Der Algorithmus findet nun folgende Mengen:

Index i	Gefundene Menge S
1	$\{1\}$
2	$\{1, 2\}$
3	$\{1, 2\}$

Der Algorithmus nimmt keine Zahlen mehr ab dem Index auf, da dann die Bedingung $\sum_{a_i \in S} a_i \leq B$ nicht mehr gelten würde, da $1 + 2 + 8 = 11 > 10$ gilt.

Das Ergebnis erfüllt somit nicht die Bedingung eines ρ -Approximationsalgorithmus für Maximierungsprobleme $L^*/L_A \leq \rho$. Stattdessen gilt für das Ergebnis $L_A = 3$, die totale Summe $L^* = B = 10$ und $\rho = 2$ die Gleichung $L^*/L_A = 10/3 = \overline{3,3} \not\leq \rho$.

Damit ist der angegebene Algorithmus kein 2-Approximationsalgorithmus.

□

b)

Algorithm 1 FindTotalSum

```

1: procedure FINDTOTALSUM( $A, B, \rho$ )
2:    $A \leftarrow \text{ConvertToList}(A)$ 
3:    $A \leftarrow \text{MergeSort}(A)$ 
4:    $T := 0$ 
5:    $S := \emptyset$ 
6:   for  $i \in \{n, \dots, 1\}$  do
7:     if  $T + a_i \leq B$  then
8:        $T \leftarrow T + a_i$ 
9:        $S \leftarrow S \cup \{a_i\}$ 
10:    end if
```

```

11:   end for
12: end procedure

```

Laufzeitbeweis

Der Algorithmus soll die Laufzeitschranke von $\mathcal{O}(n \cdot \log n)$ nicht überschreiten, was zu beweisen gilt:

Eine Menge in eine Liste zu konvertieren ist bei der Erzeugung einer verketteten Liste in linearer Laufzeit möglich.

Die Liste wird nun mittels MERGESORT sortiert. Die worst-case-Laufzeit von MERGESORT liegt dabei in $\mathcal{O}(n \cdot \log n)$.

Die Schleife (Zeile 6 bis 11) wird genau n mal ausgeführt. Alle Operationen in der Schleife lassen sich in konstanter Zeit bewerkstelligen, sofern man die Menge genügend schlaue implementiert (z.B. als verkettete Liste).

Somit liegt die Gesamtlaufzeit auch in $\mathcal{O}(n \cdot \log n)$. □

Korrektheitsbeweis

Zunächst sei das triviale ausgesprochen: Da A aufsteigend sortiert ist gilt die Ungleichung $a_i < a_{i+1}$, es gibt zudem kein Element doppelt (deswegen auch keine \leq -Relation).

Der Algorithmus überspringt zudem alle Elemente die größer als die Schranke B sind. Da diese auch nicht in L^* auftauchen können (weil $L^* \leq B$ gilt), braucht man diese auch nicht gesondert zu betrachten. Relevant wird es ab dem Element $a_k \leq B$ mit $1 \leq k \leq n$. Gibt es kein k für das die Ungleichung gilt (sprich sind alle Elemente größer als B), so ist $S = L^* = 0$.

Für die Hauptschleife (Zeile 6 bis 11) gibt es eine Schleifeninvariante: *Ist $T + a_i > B$, so wird a_i nicht aufgenommen. Findet sich kein a_j mit $0 \leq j \leq i$, für das $T + a_j \leq B$ gilt, so ist $T \geq \frac{L^*}{2}$.* Lässt sich also kein j finden ist der Algorithmus entweder zu Ende oder hat ein genügend genaues Ergebnis geliefert für das gilt $T \geq \frac{L^*}{2}$. Daraus folgt, dass der angegebene Algorithmus ein 2-Approximationsalgorithmus ist.

Beweis der Invariante mittels Widerspruch für *nicht* beendeten Algorithmus

Lässt sich ein a_j finden ist nichts zu zeigen. Es wird also nur die Situation betrachtet in der sich kein a_j finden lässt und in der der Algorithmus noch nicht zu Ende ist (also wenn $j \neq 1$). Der aktuelle Laufindex der Schleife ist dabei i .

Angenommen es lässt sich kein a_j finden, dann ist $T < \frac{L^*}{2}$.

Da sich kein a_j finden lässt gilt $T + a_j > B$ für jedes a_j mit $1 \leq j < i$. Für diese gilt dadurch $a_j > B - T \geq L^* - T > \frac{L^*}{2}$, was direkt aus $T + a_j > B$ und $T < \frac{L^*}{2}$ folgt. Somit gilt auch, dass jedes $a_k < \frac{L^*}{2}$ mit $i \leq k \leq n$ ist, da sich die bisherige Summe T aus mindestens einem a_k zusammensetzt. Es muss also $a_j > a_k$ gelten.

Die Liste aller Zahlen A ist jedoch aufsteigend sortiert, wodurch $a_j > a_k$ einen Widerspruch darstellt. Daraus folgt, dass $T \geq \frac{L^*}{2}$ gelten muss wenn sich kein a_j finden lässt.

Es fehlt nun noch der Beweis für $T \geq \frac{L^*}{2}$ wenn der Algorithmus zu Ende gekommen ist.

Beweis von $T \geq \frac{L^*}{2}$ für beendeten Algorithmus

Man kann zwei Fälle unterscheiden: Entweder das letzte Element a_1 wurde aufgenommen oder nicht aufgenommen. Wurde es nicht aufgenommen, so folgt aus obigem Beweis, dass $T \geq \frac{L^*}{2}$ gilt und es ist nichts zu zeigen.

Wenn a_1 aufgenommen wurde, ...

Aufgabe 6.2