28.04.08

Problem 1.11 (Knapsack)

Gegeben: n Objekte mit Größe $l_1, \dots, l_n$
Wert $c_1, \dots, c_n$,
Container der Größe L.

Gesucht: Teilmenge $I \subseteq \{1, \dots, n\}$

mit $\sum_{i \in I} l_i \leq L$

und $\sum_{i \in I} c_i$ möglichst groß.

(Also: Möglichst wertvolle Auswahl, die in den Container passt!)

Korollar 1.12

Knapsack ist NP-vollständig

Beweis:

Reduktion von Partition:

Setze $l_i = c_i$ und teste, ob es eine Lösung des Wertes L gibt.

□

Kapitel 2 : Verfahren für eindimensionale Probleme

2.1 Dynamische Programmierung für SUBSET SUM

Zurück zu Partition:

Beispiel 2.1

Gegeben : $4, 4, 8, 10, 14, 18 \rightarrow \sum_{i=1}^{6} \ell_i = 58 = 2L$

Gesucht : $i \in S$ mit $\sum_{i \in S} \ell_i = L$

Beobachtung: Nicht lösbar, da nur gerade Zahlen erzielbar sind!

Frage: Welche Zahlen sind erzielbar? → SUBSET SUM

Probieren:

~~1~~	4	~~6~~	8	10	12	14	16	18	20 (?!)	22	24	26	28
~~56~~	54	~~52~~	50	48	46	44	42	40	38	36	34	32	30

Wie zeigt man, dass 20 NICHT erzielbar ist?

Antwort:

- 18 kann nicht verwendet werden, sonst würde man eine 2 brauchen.
- 14 kann nicht verwendet werden, sonst würde man eine 6 brauchen (das geht mit dem Rest nicht!)
- 10 kann nicht verwendet werden, sonst müsste man mit dem Rest (4,4,8) eine 10 erzielen.
- Der Rest hat Summe 16.

Beobachtung:

Man argumentiert "rückwärts", was nicht geht.
Man kann auch "vorwärts" argumentieren, was geht!

Idee:

Führe Buch über die Zahlen, die mit den ersten i Zahlen $l_1, \dots, l_i$ erzielbar sind!

$$E(i,z) = \begin{cases} 1 & \text{falls } z \text{ aus } l_1, \dots, l_i \text{ erzielbar} \\ 0 & \text{falls } z \text{ nicht erzielbar aus } l_1, \dots, l_i \end{cases}$$

Dann gilt natürlich:

$E(0,z) = 0$ für alle $z \in [1, L]$

und

$E(i,z) = 1$ wenn: $E(i-1,z) = 1$ → es geht ohne l_i

oder $E(i-1, z-l_i) = 1$ → es geht mit l_i.

Systematisch liefert das

In Vorlesung: Zuerst das Beispiel von S.14

Algorithmus 2.2 (Dynamic Programming für SUBSET SUM)

Eingabe: Ganzzahlen $l_1, \dots, l_n$, Intervallgrenze L

Ausgabe: Funktion $E: [1,L] \to \{0,1\}$

$z \mapsto e_z$,

die beschreibt, welche Zahlen erstellbar sind.

(1)
```
E(0,0)=1;
FOR z= 1 TO L
    E(0,z) = 0
```

(2)
```
FOR i=1 TO n {
  FOR z=1 TO L {
    IF ( (E(i,z)=1) OR (E(i-1, z-l_i)=1) )
        E(i,z)=1
  }
}
```

FOR z=1 TO L

e(z) = E(n,z)

In Vorlesung: zuerst das Beispiel

Am Beispiel:

i \ z	2	4	6	8	10	12	14	16	18	20	22	24	26	28
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	1	0	0	0	0	0	0	0	0	0	0
3	0	1	0	1	0	1	0	1	0	0	0	0	0	0
4	0	1	0	1	1	1	1	1	1	0	1	0	1	0
5	1	0	1	0	1	1	1	1	1	0	1	1	1	1
6	1	0	1	0	1	1	1	1	1	0	1	1	1	1

Satz 2.3

Algorithmus 2.2 löst SUBSET SUM.
Die Komplexität ist $O(nL)$.

Beweis:

Klar nach Vordiskussion!

Frage 2.4

Wenn wir SUBSET SUM in $O(nL)$ lösen können – warum ist dann nicht $P = NP$?
Wir haben doch ein NP-schweres Problem gelöst?!

Antwort:

$O(nL)$ ist nicht polynomiell!

L ist eine Zahl, für die wir nur $\log L$ als Inputlänge brauchen

– und L ist NICHT polynomiell beschränkt durch $\log L$!

Trotzdem ist das besser als nichts...

Definition 2.5

Ein Algorithmus heißt pseudopolynomiell, wenn seine Laufzeit durch ein Polynom der Inputlänge und der größten Eingabezahl begrenzt ist.

2.2 Dynamic Programming für KNAPSACK

Zur Erinnerung:

Gegeben: n Gegenstände mit Größe $\ell_1, \dots, \ell_n$
Wert $c_1, \dots, c_n$,
Container mit Größe L.

Gesucht: $S \subseteq \{1, \dots, n\}$ mit

$$\sum_{i \in S} \ell_i \leq L$$

und $\sum_{i \in S} c_i$ größtmöglich

Hier funktioniert eine ähnliche Idee wie für SUBSET SUM!
Wir berechnen die erzielbaren Wertzahlen, wobei wir die Kapazitätsgrenze berücksichtigen.

Mit $C := \sum_{i=1}^{n} c_i$ erhalten wir

den folgenden Algorithmus:

Algorithmus 2.6 (Dynamic Programming für Knapsack)

Eingabe: Positive ganze Zahlen
$l_1, \dots, l_n, c_1, \dots, c_n, L$

Ausgabe: Eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit
$\sum_{i \in S} l_i \leq L$ und
$\sum_{i \in S} c_i$ größtmöglich.

(1) $x(0,0) := 0$; FOR $k=1$ TO L $\{ x(0,k) := \infty \}$

(2) FOR $j := 1$ TO n DO {
FOR $k := 0$ TO C DO {
$s(i,k) := 0$ und $x(j,k) := x(j-1,k)$
}
FOR $k := c_j$ TO C DO {
IF $x(j-1, k-c_j) + l_j \leq \min\{L, x(j,k)\}$ THEN
$x(j,k) := x(j-1, k-c_j) + l_j$ und $s(j,k) := 1$
}

(3) Sei $k = \max\{i \in \{0, \dots, C\} : x(n,i) < \infty\}$. Setze $S := \emptyset$.
FOR $j := n$ DOWN TO 1 DO {
IF $s(j,k) = 1$ THEN {
$S := S \cup \{j\}$ und $k := k - c_j$
}
}