Lösungsstrategien für NP-schwere Probleme der Kombinatorischen Optimierung

— Übungsblatt 5 —

Walter Stieben (4stieben@inf)
Tim Reipschläger (4reipsch@inf)
Louis Kobras (4kobras@inf)
Hauke Stieler (4stieler@inf)

Abgabe am: 23. Mai 2016

Aufgabe 5.1

Aufgabe 5.2

a)

Laufzeitanalyse

Die Laufzeit des von `EXPLORE`(Φ, d) lässt sich mittels Rekurrenzgleichung und Substitutionsmethode bestimmen.

Zunächst die Rekurrenzgleichung in Abhängigkeit von d und n. Die Menge C taucht nicht auf, da sie nicht Teil der Eingabe von `EXPLORE`(Φ, d) ist. Dem n wäre also noch ein linearer Faktor hinzu zu zählen, was jedoch wenig von Bedeutung ist.

$$T(d,n) = \begin{cases} n & \text{, für } d = 0 \\ n \cdot 3 \cdot T(d-1) & \text{, sonst} \end{cases}$$

Mittels der Sibstitutionsmethode ergibt sich folgende Kette:

$$\begin{aligned}
T(d,n) &= n + 3 \cdot T(d-1) \\
&= n + 3 \cdot (n + 3 \cdot T(d-2)) \\
&= n + 3n + 9 \cdot T(d-2) \\
&= 4n + 9 \cdot T(d-2) \\
&= 4n + 9 \cdot (n + 3 \cdot T(d-3)) \\
&= 4n + 9n + 27 \cdot T(d-3) \\
&= 13n + 27 \cdot T(d-3) \\
&= 13n + 27 \cdot (n + 3 \cdot T(d-4)) \\
&= 13n + 27n + 81 \cdot T(d-4) \\
&= 40n + 81 \cdot T(d-4) \qquad (1) \\
&\vdots \\
&= \frac{3^k - 1}{2} + 3^k \cdot T(d-k)
\end{aligned}$$

Nach drei Substitutionen kann man erkennen, der Vorfaktor von T stets 3^k ist $(3, 9, 27, 81, \dots)$ und der Faktor vom n stets $\frac{3^k-1}{2}$ sein muss $(1, 4, 13, 40, \dots)$.

> Beweise zu den Gleichungen ersparen wir uns hier der einfachheitshalber.

Der Abbruch des Algorithmus' geschieht bei $k = d$. Dadurch ist $T(d-k) = T(d-d) = T(0) = n$ und wir erhalten die finale Laufzeit folgender Größenordnung:

$$\mathcal{O}\left(\frac{3^d - 1}{2} + 3^d \cdot n\right)$$

b)

Wenn man $d = \frac{n}{2}$ wählt, so erhält und in die Laufzeit aus 2.a) einsetzt, erhält man folgende neue Laufzeit:

$$\begin{aligned}
\frac{3^{n/2} - 1}{2} + 3^{n/2} \cdot n &= 3^{n/2} \cdot 0,5 - \left(\frac{1}{2}\right) + 3^{n/2} \cdot n \qquad \text{(Die } \mathcal{O}\text{-Notation ignoriert Konstanten)} \\
&= 3^{n/2} + 3^{n/2} \cdot n \\
&= \left(\sqrt{3}\right)^n + \left(\sqrt{3}\right)^n \cdot n \\
&= (n+1) \cdot \left(\sqrt{3}\right)^n
\end{aligned}$$

> Die Notation $n/2$ statt $\frac{n}{2}$ wird zwar von Mathematikern und Theoretikern oftmals abgelehnt, wird hier jedoch zugunsten der Lesbarkeit verwendet.

Somit ist die Laufzeit von `EXPLORE`(Φ, d) mit $d = \frac{n}{2}$ in $\mathcal{O}\left((n+1) \cdot \left(\sqrt{3}\right)^n\right)$.

Um jedoch alle möglichen Belegungen durch zu gehen bedarf es $\frac{2^n}{\frac{n}{2}} = \frac{2^{n+1}}{n}$ Aufrufe. Jeder Aufruf von `EXPLORE`(Φ, d) geht $d = \frac{n}{2}$ Belegungen durch, wodurch man nicht volle 2^n mögliche Belegungen benötigt.

Die Anzahl multipliziert mit der Anzahl an Schritten aus vorheriger Rechnung ergibt die Gesamtlaufzeit von $\mathcal{O}\left(\frac{2^{n+1}}{n} \cdot (n+1) \cdot \left(\sqrt{3}\right)^n\right)$.

Somit haben wir eine Laufzeit der Form $\mathcal{O}\left(p(n) \cdot \left(\sqrt{3}\right)^n\right)$.