DAP2 – Präsenzübung 5

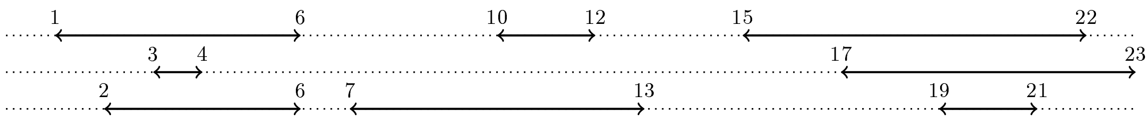
Besprechung: 24.05.2017 — 26.05.2017

Präsenzaufgabe 5.1: (Gierige Algorithmen)

Die Fakultät Informatik bietet einen Help-Desk-Service, der von n Tutorinnen angeboten wird. Die Tutorin T_i, $1 \leq i \leq n$ ist im Dienst im Zeitintervall $[a_i, b_i]$, wobei a_i und b_i positive reelle Zahlen sind. Sei $S = \{[a_1, b_1], \ldots, [a_n, b_n]\}$ die Menge von Intervallen, in denen die Tutorinnen im Dienst sind.

Einige von diesen Tutorinnen sollten als verantwortliche Tutorinnen eingestellt werden, und damit besser bezahlt werden. Sei V die Menge der Dienstzeitintervalle der verantwortlichen Tutorinnen, $V \subseteq S$. Für jede Tutorin T_i mit Dienstzeitintervall $[a, b] \in S$ muss eine verantwortliche Tutorin T_j mit Dienstzeitintervall $[c, d] \in V$ existieren, sodass $[a, b] \subseteq [c, d]$ ist (d.h., es gilt $c \leq a$ und $b \leq d$).

Z.B. stellen für die unten angegebenen Dienstzeitintervalle der n Tutorinnen die Intervalle $V = \{[1, 6], [7, 13], [15, 22], [17, 23]\}$ eine Auswahl der verantwortlichen Tutorinnen dar.



a) Beschreiben Sie einen *gierigen* Algorithmus, der bei Eingabe einer Menge S von Dienstzeitintervallen eine minimale Menge der verantwortlichen Tutorinnen V berechnet (d.h., V enthält minimal viele Intervalle). Geben Sie den Algorithmus auch in Pseudocode an.

Lösung:

Wir sagen dass die Menge der verantwortlichen Tutorinnen V die Menge aller Tutorinnen S überdeckt, d.h. es gibt für jede Tutorin mit dem Dienstzeitintervall $[a, b]$ eine verantwortliche Tutorin, deren Dienstzeitintervall $[c, d]$ das Intervall $[a, b]$ enthält/überdeckt.

Zuerst sortieren wir alle Intervalle aufsteigend bzgl. ihrer linken Intervallendpunkte a_i. Angenommen es gilt $a_1 \leq a_2 \leq \ldots \leq a_n$. Wir initialisieren $V = \emptyset$. Solange wir noch nicht überdeckte Intervalle haben, suchen wir das Intervall mit dem kleinsten linken Endpunkt a_j. Falls es mehrere mögliche Intervalle gibt, wählen wir das mit dem größten rechten Endpunkt. Dieses Intervall fügen wir V hinzu und wiederholen diesen Vorgang bis alle Intervalle überdeckt sind. Der Pseudocode sieht folgendermaßen aus.

TutorinnenAuswahl(Sortierte Intevalle S):

```
i = 1
V = ∅
while i ≤ n do
    s = a_i
    max = i
    while s = a_i do
        if b_max < b_i then
            max = i
        i = i + 1
    V = V ∪ {[a_max, b_max]}
    while b_i < b_max do
        i = i + 1
return V
```

b) Analysieren Sie die Laufzeit Ihres Algorithmus.

Lösung:
Der Algorithmus hat Laufzeit $O(n)$, wenn die Intervalle bereits sortiert sind. Um die Intervalle zu sortieren, benötigen wir zusätzlich $O(n \log n)$ Zeit.

c) Beweisen Sie, dass Ihr Algorithmus eine optimale Lösung berechnet.

Lösung:

Wir dürfen annehmen, dass die Sortierung der Intervalle $[a_i, b_i]$ am Anfang nach zwei Kriterien durchgeführt wurde: zuerst aufsteigend nach den Anfangspunkten a_i und falls Intervalle den gleichen Anfangspunkt haben, dann noch absteigend nach Endpunkten b_i sortieren. Dies ist äquivalent wie in der obigen Lösung.

Nehmen wir an, die Lösung V des gierigen Algorithmus ist nicht optimal, d.h. es gibt eine Überdeckung $V_{OPT} \subseteq S$ mit $|V_{OPT}| < |V|$. Da V_{OPT} auch die Intervalle aus V überdecken muss, muss in V_{OPT} ein Intervall $[a', b']$ existieren, so dass $[a_i, b_i] \subseteq [a', b']$ und $[a_j, b_j] \subseteq [a', b']$ ist, mit $a_i < a_j$, $[a_i, b_i] \in V$ und $[a_j, b_j] \in V$.

Es können zwei Fälle vorkommen: (1) $a_i > a'$ oder (2) $a_i = a'$ (da $a_i < a'$ ausgeschlossen ist, weil $[a_i, b_i] \subseteq [a', b']$). Wegen unsere Auswahlstrategie gilt auch, dass $b_i < b_j$ ist. Wegen Überdeckungseigenschaften gilt es, dass $b_i \leq b'$ und $b_j \leq b'$.

- Falls $a_i = a'$, dann hätten wir nach unserer gierigen Strategie $[a', b']$ anstatt $[a_i, b_i]$ ausgewählt, da $b_i < b_j \leq b'$ ist.
- Fall $a_i > a'$: Da $b' \geq b_j > b_i$ gilt, ist zum Zeitpunkt der Auswahl von $[a_i, b_i]$ im gierigen Algorithmus zumindest das Teilintervall $[b_i, b']$ des Intervalls $[a', b']$ nicht überdeckt. Somit müssten wir $[a', b']$ wieder in die gierige Lösung auswählen, da $[a', b']$ bei der Auswahl von $[a_i, b_i]$ nicht überdeckt war und einen kleineren Anfangspunkt $a' < a_i$ hat.

Somit kann kein Intervall aus V_{OPT} zwei Intervalle aus V überdecken, und $|V_{OPT}| \geq |V|$. Damit ist die gierige Lösung optimal.

c') (alternativer Beweis) Wir nehmen immer noch an, dass die Intervalle nach ihren Anfangspunkten sortiert sind, d.h. $a_1 \leq a_2 \leq \ldots, \leq a_n$.

Angenommen die Lösung V des gierigen Algorithmus ist nicht optimal, d.h. es gibt eine Überdeckung $V_{OPT} \subseteq S$ mit $|V_{OPT}| < |V|$. Seien $i_1, \ldots, i_k$ mit $k = |V|$ die ausgewählten Intervalle in V und $j_1, \ldots, j_l$ mit $l = |V_{OPT}|$ die Intervalle der optimalen Lösung. O.B.d.A. gilt $a_{i_1} \leq a_{i_2} \leq \ldots \leq a_{i_k}$ und $a_{j_1} \leq a_{j_2} \leq \ldots \leq a_{j_l}$.

Hilfsbehauptung: Für alle $m \in \{1, \ldots, l\}$ gilt $b_{i_m} \geq b_{j_m}$.
Beweis: Wir beweisen die Behauptung per Induktion über m.

I.A. $m = 1$: Da V_{OPT} auch $[a_1, b_1]$ überdecken muss, gilt $a_{j_1} = a_1 = a_{i_1}$. Nach unserer gierigen Wahl gilt somit $b_{i_1} \geq b_{j_1}$.

I.V. Für ein beliebiges aber festes $m \in \{2, \ldots, l\}$ gilt die Behauptung für alle $m' < m$.

I.S. Da i_m das nächste (bzgl. den Anfangspunkten) nicht von den Intervallen $i_1, \ldots, i_{m-1}$ überdeckte Intervall ist und nach I.V. $b_{i_{m'}} \geq b_{j_{m'}}$ für alle $m' < m$ gilt, ist auch i_m nicht von den Intervallen $j_1, \ldots, j_{m-1}$ überdeckt. Damit muss $a_{j_m} = a_{i_m}$ gelten, da sonst das Intervall i_m nicht von V_{OPT} überdeckt werden kann. Nach unserer gierigen Wahl gilt somit $b_{i_m} \geq b_{j_m}$.

Da $k > l$ ist, gibt es ein Intervall i_{l+1}, das von den Intervallen $i_1, \ldots, i_l$ nicht überdeckt wird. Nach der Hilfsbehauptung gilt $b_{i_m} \geq b_{j_m}$ für alle $1 \leq m \leq l$. Somit kann das Intervall i_{l+1} auch nicht von den Intervallen $j_1, \ldots, j_l$ überdeckt werden. Dies ist ein Widerspruch zur Gültigkeit der optimalen Lösung. Somit gilt $l = k$.

□

Präsenzaufgabe 5.2: (Gierig ist nicht immer gut)

Zu den bekanntesten und zugleich wichtigsten Optimierungsproblemen zählt das Rundreiseproblem (Traveling Salesman Problem, kurz TSP):

Eingabe: Orte $1, \ldots, n$, sowie die Kosten $c(i, j)$ für $1 \leq i, j \leq n$ mit $i \neq j$, um von i nach j zu reisen.

Ziel: Eine kostenminimale Rundreise, die o.B.d.A. im Ort 1 beginnt und wieder endet und bei der jeder Ort $2 \leq i \leq n$ genau einmal besucht wird. Formal heißt dies, dass eine Permutation π auf $\{1, \ldots, n\}$ mit Fixpunkt $\pi(1) = 1$ gesucht wird, die die Summe

$$c(\pi(n), \pi(1)) + \sum_{i=1}^{n-1} c(\pi(i), \pi(i+1))$$

minimiert.

Gehen Sie davon aus, dass $c(i, j) = c(j, i)$ für alle $i \neq j$ gilt, dass also nur symmetrische Instanzen zugelassen sind, was beim allgemeinen TSP nicht gefordert wird.

a) Welche Laufzeit hat ein Algorithmus, der die kostenminimale Rundreise mittels vollständiger Suche bestimmt, d.h., indem er alle möglichen Rundreisen ausprobiert?

Lösung: Es gibt $(n-1)!$ verschiedene mögliche Rundreisen. Für jede Rundreise braucht man $O(n)$ Zeit, um die Kosten zu bestimmen. Daher braucht der Algorithmus insgesamt $O(n!)$ Zeit.

b) Eine gierige Strategie zur Berechnung einer Rundreise würde darin bestehen, dass wir in Ort 1 startend zunächst einen Ort j wählen, für den $c(1,j) = \min\{c(1,k) \mid 2 \leq k \leq n\}$. Von j ausgehend wird dann der nächste Ort l, verschieden von den schon besuchten Orten, so bestimmt, dass $c(j,l)$ unter allen noch nicht besuchten Orten l einen minimalen Wert hat. Dieses Verfahren wird fortgesetzt, bis alle Orte besucht sind.

Zeigen Sie, dass die gierige Strategie beliebig schlechte Lösungen liefern kann, d.h. zeigen Sie, dass es zu jedem $k > 1$ eine Instanz P für das Rundreiseproblem gibt, sodass

$$\frac{cost_{\text{greedy}}(P)}{cost_{\text{opt}}(P)} > k.$$

Hierbei bezeichnen $cost_{\text{greedy}}(P)$ die Kosten einer Rundreise, die mit der gierigen Strategie bestimmt wird und $cost_{\text{opt}}(P)$ die Kosten einer optimalen Rundreise.

Hinweis: Versuchen Sie die gierige Strategie in eine Falle zu locken, von der nur noch teure Kanten zu noch unbesuchten Orten wegführen!

Lösung: Wir konstruieren eine Instanz P, welche den greedy Algorithmus auf einen bestimmten *kostengünstigen* Pfad $1 \longrightarrow 2 \longrightarrow \ldots \longrightarrow n-1 \longrightarrow n$ lockt. Wir wählen für alle $1 \leq i \leq n-1$ die Kosten $c(i, i+1) = 1$. Einmal im Ort n angekommen, bleibt dem Algorithmus nichts anderes übrig, als zurück zum Ausgangspunkt zu reisen. Dieser Kante weisen wir einen *hohen* Kostenwert $c(n,1) = M$ zu, dessen Wahl noch näher zu bestimmen ist. Allen anderen Kanten weisen wir Kosten 2 zu. Es gilt nun offensichtlich $cost_{\text{greedy}}(P) = n - 1 + M$ und $cost_{\text{opt}}(P) = n + 2$. Eine optimale Tour wäre z.B.: $\pi = (1, 2, 3, \ldots, n-2, n, n-1)$.

Zu zeigen bleibt, dass es ein geeignetes M gibt, sodass

$$\frac{cost_{\text{greedy}}(P)}{cost_{\text{opt}}(P)} = \frac{n-1+M}{n+2} > k$$

gilt.

Wir formen äquivalent nach M um:

$$\begin{aligned} & \frac{n-1+M}{n+2} > k \\ \Leftrightarrow\quad & n - 1 + M > k \cdot (n+2) \\ \Leftrightarrow\quad & M > k \cdot (n+2) + 1 - n \end{aligned}$$

□