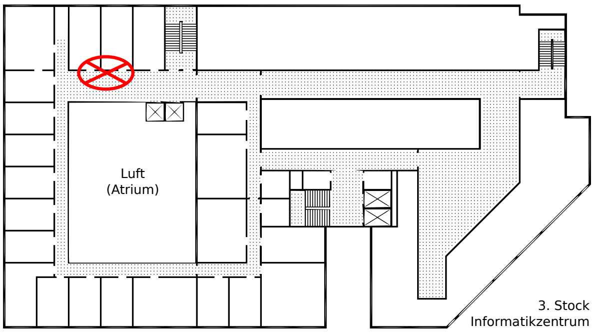
Abteilung Algorithmik **Sommer 2015**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Dr. Christian Scheffer

Algorithmen und Datenstrukturen II
Übung 4 vom 10. 06. 2015

Abgabe der Lösungen bis zum Montag, den 22. 06. 2015 um 14:00 im Hausaufgabenrückgabeschrank.



Bitte die Blätter vorne deutlich mit eigenem Namen sowie Matrikel- und Gruppennummer versehen!

Aufgabe 1 (DP für Matrizenmultiplikation): Gegeben sei eine Sequenz von Matrizen $\langle M_1, ..., M_n \rangle$. Für $i = 1, ..., n$ sei M_i eine $d_i \times d_{i+1}$-Matrix. Somit ist die Matrixmultiplikation $M_1 \cdot M_2 \cdot ... \cdot M_n$ wohldefiniert.

Anhand des folgendem Beispiels kann man erkennen, dass die Klammerung des Ausdrucks $M_1 \cdot M_2 \cdot ... \cdot M_n$ einen Einfluss auf die Anzahl der Multiplikationen von einzelnen Einträgen hat: $M_1 \in \mathbb{R}^5 \times \mathbb{R}^5$, $M_2 \in \mathbb{R}^5 \times \mathbb{R}^{100}$ und $M_3 \in \mathbb{R}^{100} \times \mathbb{R}^3$. Der Ausdruck $(M_1 \cdot M_2) \cdot M_3$ verursacht somit $5 \cdot 5 \cdot 100 + 5 \cdot 100 \cdot 3$ Multiplikationen, während $M_1 \cdot (M_2 \cdot M_3)$ nur $5 \cdot 100 \cdot 3 + 5 \cdot 5 \cdot 3$ Multiplikationen verursacht.

Entwirf einen Algorithmus, der mittels dynamischer Programmierung berechnet, wie viele Einzelmultiplikationen bei einer optimalen Klammerung nötig sind.

Verwende hierzu folgenden Ansatz: Für ein $k \in \{1, ..., n-1\}$ ergibt sich $(M_1 \cdot ...M_k) \cdot (M_{k+1} \cdot ... \cdot M_n)$. Die damit nötigen Multiplikationen sind die, die für die Berechnung von $M_1 \cdot ... \cdot M_k = M$, $M_{k+1} \cdot ... \cdot M_n =: M'$ und $M \cdot M'$ nötig sind.

Hinweise zur Matrizenmultiplikation, siehe 4. große Übung. **(10 Punkte)**

Aufgabe 2 (Approximation von Set Cover): Gegeben sind eine endliche Menge U und eine Familie $\mathcal{F}$ von Teilmengen von U. $C \subseteq U$ *überdeckt* alle Elemente in C. Ein *Set Cover (SC)* von $(U, \mathcal{F})$ ist eine Auswahl der Mengen in $\mathcal{F}$, $F = \{F_1, \ldots, F_m\} \subseteq \mathcal{F}$, die U überdeckt: $U = \bigcup_{i=1}^{m} F_i$. Gesucht ist ein Set Cover kleinstmöglicher Kardinalität. Es darf angenommen werden, dass $\mathcal{F}$ ein Set Cover von $(U, \mathcal{F})$ ist.

Algorithmus 1 ist ein Greedy-Approximationsalgorithmus für SC, der im Folgenden genauer untersucht werden soll. Dabei dienen Zeilen 7 und 8 lediglich der Analyse und spielen für die Korrektheit von Algorithmus 1 keine Rolle.

Algorithmus 1 Greedy Approximation von Set Cover

function $\textsc{GreedySC}(U, \mathcal{F})$
    $C \leftarrow \emptyset$
    $\bar{C} \leftarrow U$
    $\text{SC} \leftarrow \emptyset$

    while $C \neq U$ **do**
        $S \leftarrow \max_{S \in \mathcal{F}} \left| S \cap \bar{C} \right|$

        $\alpha \leftarrow 1/|S \cap \bar{C}|$
        $\forall s \in S \cap \bar{C} : \quad \text{price}(s) \leftarrow \alpha$

        $C \leftarrow C \cup S$
        $\bar{C} \leftarrow \bar{C} \setminus S$
        $\text{SC} \leftarrow \text{SC} \cup \{S\}$

    return SC

Sei OPT die Größe eines kleinsten Set Covers von $(U, \mathcal{F})$.

a) Konstruiere ein Beispiel $(U, \mathcal{F})$, dessen optimale Lösung mindestens zwei Teilmengen $F_1, F_2 \in \mathcal{F}$ benötigt und bei dem Algorithmus 1 nicht die optimale Lösung ausgibt.

b) Wende Algorithmus 1 auf Dein Beispiel an. Gib dabei für jede Iteration S, $S \cap \bar{C}$, α, C sowie $\bar{C}$ an. Führe nach Ablauf des Algorithmus für alle $e \in S$ price(e) auf.

c) Zeige: Algorithmus 1 ist korrekt, gibt also immer ein Set Cover aus und terminiert.

d) Zeige: In jeder Iteration von Algorithmus 1 gilt $\alpha \leq \text{OPT}/|\bar{C}|$. (Hinweis: In jeder Iteration kann $\bar{C}$ mit $\leq$ OPT Elementen überdeckt werden.)

e) Sei $U = \{e_1, \ldots, e_n\}$, wobei die Elemente in der Reihenfolge aufgelistet seien, in der Algorithmus 1 sie in C einfügt. Folgere aus der letzten Teilaufgabe: Für alle k gilt $\text{price}(e_k) \leq \frac{\text{OPT}}{n-k+1}$.

f) Folgere: Algorithmus 1 liefert eine H_n-Approximation von Set Cover, wobei $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$. (Hinweis: Verteile die Kosten des gefundenen Set Covers mit Hilfe von price(e_i) auf U.)

(1+3+2+2+1+1 Punkte)