TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 22.01.2015
Version: 2015-01-20 15:13

11. Übungsblatt für alle Tracks zur Vorlesung Entwurf und Analyse von Algorithmen, WS 14/15

Abgabe: *Bis Freitag, 30.01.2015, 12:00 Uhr, Kasten im Treppenhaus 48-6.*

Basisaufgaben

B11.1: Einordung Entwurfsmethoden

3 Punkte

Geben Sie für die folgenden Algorithmen aus der Vorlesung an, welchem Algorithmenentwurfsmuster – Divide & Conquer (D&C), Dynamische Programmierung (DP) oder Greedy-Algorithmen – sie folgen.

Begründen Sie Ihre Antwort und argumentieren Sie jeweils kurz, warum die Vorausetzungen des jeweiligen Entwurfsmusters hier erfüllt sind; also bei Divide & Conquer, warum die Teilprobleme unabhängig gelöst werden können, bei Dynamischer Programmierung, warum das Bellmansche Optimalitätskriterium gilt, und bei Greedy-Algorithmen, warum gewählte Teile der Lösung nie revidiert werden müssen.

a) Berechnung maximaler Teilsummen, Optimierung 1 (mit quadratischer Laufzeit, siehe Buch Seite 24 Mitte)

b) Berechnung maximaler Teilsummen, Optimierung 2 (mit Laufzeit $\Theta(n \log n)$, siehe Buch Seite 24 unten)

c) Kruskals Algorithmus für MSTs

d) Bellman-Ford (siehe Aufgabe A9.1)

e) Selectionsort

f) Quicksort

g) (2-Wege-) Mergesort (siehe Buch Abschnitt 5.4.1)

B11.2: Optimale statische BSTs I

3 Punkte

Seien für $n \in \mathbb{N}$ die Schlüsselmenge $A_n = \{1, \ldots, n\}$ und beliebige, aber feste Zugriffswahrscheinlichkeiten $P = (p_1, \ldots, p_n) \in [0,1]^n$ mit $\sum_{i=1}^{n} p_i = 1$ gegeben. Wir nennen einen Suchbaum T für A (genau) dann optimal für P, wenn er unter allen Suchbäumen für A die erwarteten Suchkosten $\sum_{i=1}^{n} p_i \cdot \mathrm{niv}_T(i)$ minimiert.

Begründen Sie, warum das Bellmansche Optimalitätskriterium für dieses Problem erfüllt ist, sich die dynamische Programmierung also zur Optimierung eignet.

B11.3: Optimale statische BSTs II

3 Punkte

Wir betrachten nochmal das Szenario aus B11.2.

Entwerfen Sie mittels dynamischer Programmierung einen Algorithmus, der optimale binäre Suchbäume konstruiert.

Begründen Sie die Korrektheit und bestimmen Sie die asymptotische Worst-Case Laufzeit (als Θ-Klasse) Ihres Algorithmus.

B11.4: Matroide

3 Punkte

Sei S eine endliche Menge mit $|S| \geq 2$. Seien weiterhin mit $\{S_1, \ldots, S_k\}$ eine Partition von S und mit $\mathcal{U} = \{A \subseteq S \mid |A \cap S_i| \leq 1, 1 \leq i \leq k\}$ ein Universum gegeben.

Zeigen Sie: $(S, \mathcal{U})$ ist ein Matroid.

Aufbauaufgaben

A11.1: Exakte Lösung für TSP

5 Punkte

Entwerfen Sie einen Algorithmus, der das Travelling-Salesman-Problem (kurz TSP, siehe Buch Definition 8.13, Seite 343) in Zeit $\mathcal{O}(2^n \cdot p(n))$ für ein Polynom p (exakt) löst.

Tipp: Versuchen Sie es mit einem induktiven Algorithmus, der mögliche Routen stückweise verlängert. Versuchen Sie nicht, Platz zu sparen!

A11.2: Münzwechselproblem

3 + 3 Punkte

Das *Münzwechselproblem* ist das Problem, einen gegebenen Betrag an (Wechsel-)Geld mit möglichst wenig Münzen auszuzahlen. Formal definieren wir das Problem wie folgt.

> Gegeben einen festen Satz Münzen $M = \{w_1, \ldots, w_k\}$ mit ganzzahligen Wertigkeiten $1 = w_1 < w_2 < \cdots < w_k$ und einen Zielbetrag $n \in \mathbb{N}$, bestimme $C = (c_1, \ldots, c_k)$ so, dass
>
> i) $\sum_{i=1}^{k} c_i w_i = n$ und
>
> ii) C unter allen Lösungen, die i) erfüllen, $\sum_{i=1}^{k} c_i$ minimiert.

a) Entwerfen Sie einen Greedy-Algorithmus, der das Münzwechselproblem für den Euro-Münzsatz (1, 2, 5, 10, 20, 50, 100 und 200 Cent), löst.

   Begründen Sie die Korrektheit des Algorithmus.

b) Gibt es Münzsätze, für die Greedy-Algorithmen an der Berechnung einer optimalen Stückelung scheitern können? Nehmen Sie zur Beantwortung dieser Frage an, dass jeder mögliche Satz eine Münze mit Wertigkeit 1 enthält.

A11.3: Lineare Unabhängigkeit und Matroide

3 Punkte

Sei V ein Vektorraum, $M \subset V$ eine endliche Teilmenge dessen und

$$\mathrm{Ind}_M = \{A \subseteq M \mid A \text{ linear unabhängig }\}$$

die Menge aller linear unabhängigen Mengen von Vektoren aus M.

Zeigen Sie: (M, Ind_M) ist (stets) ein Matroid.