Technische Universität
Kaiserslautern

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 23.01.2014
Version: 2014-01-28 13:42

13. Übungsblatt für Track ε zur Vorlesung Entwurf und Analyse von Algorithmen, WS 13/14

Abgabe: *Bis* ***Freitag****, 31.01.2014, 12:00 Uhr, Kasten im Treppenhaus 48-6.*

So Sie noch die Zwischenklausur für Ihre Zulassung bestehen müssen, denken Sie bitte an die Anmeldung zur 2. Zwischenklausur (siehe OLAT bzw. unsere Website), die bis zum 19.02.2014 offen ist.

Weiterhin möchten wir Sie an die Vorlesungsumfrage erinnern! Ihr ehrliches, konstruktives Feedback ist uns wichtig. Bisher haben nur Wenige teilgenommen; bitte ändern Sie das!

Falls Sie noch kein Token für den Login haben, schauen Sie bitte im SCI (48-226) vorbei.

Der Fehler der Woche

Longest Path ist in $\mathcal{NP}$, da [...] exponentielle Laufzeit erforderlich ist.

Was ist falsch?

Basisaufgaben

B1: Matroide

2 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.

B2: Dynamische Programmierung

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.

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

b) 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.

B3: co-$\mathcal{NP}$

3 Punkte

Wir definieren co-$\mathcal{NP}$ als die Menge all jener Sprachen L, deren Komplemente zur Klasse $\mathcal{NP}$ gehören, also

$$\text{co-}\mathcal{NP} := \{L \in \Sigma^* \mid (\Sigma^* \setminus L) \in \mathcal{NP}\} .$$

a) Zeigen Sie: $\mathcal{P} \subseteq \text{co-}\mathcal{NP}$.

b) Zeigen Sie: $\mathcal{NP} \neq \text{co-}\mathcal{NP} \implies \mathcal{P} \neq \mathcal{NP}$.

B4: $\mathcal{NP}$-Vollständigkeit

4 Punkte

Wir erweitern unseren Katalog $\mathcal{NP}$-vollständiger Probleme um zwei weitere:

- **Rucksack**

  Eingabe: Natürliche Zahlen $a_1, a_2, \ldots, a_k, b \in \mathbb{N}$.

  Frage: Gibt es $I \subseteq [1..k]$ mit $\sum_{i \in I} a_i = b$?

- **Partition**

  Eingabe: Natürliche Zahlen $c_1, c_2, \ldots, c_k \in \mathbb{N}$.

  Frage: Gibt es $J \subseteq [1..k]$ mit $\sum_{i \in J} c_i = \sum_{i \in [1..k] \setminus J} c_i$?

Beweisen Sie, dass Partition $\mathcal{NP}$-hart ist, wenn Rucksack $\mathcal{NP}$-vollständig ist.

Aufbauaufgaben

41. Aufgabe

2 + 1 Punkte

Wir betrachten noch einmal die Funktion B_s aus Aufgabe 33 von Blatt 11.

a) Geben Sie einen Algorithmus an, der B_s für beliebige Graphen (und s) in Polynomialzeit berechnet, das heißt

$$[B_s(n,1),\ldots,B_s(n,n)]$$

zurückgibt. Bestimmen Sie die Laufzeit Ihres Algorithmus.

b) Können Sie Ihren Algorithmus aus a) so modifizieren, dass er ohne (asymptotischen) Mehraufwand *feststellt*, ob G negative Kreise enthält?

42. Aufgabe

5 Punkte

Gegeben ist eine Menge von n Punkten in der reellen Ebene. Entwerfen Sie einen Algorithmus mit einer Laufzeit in $o(n^2)$ (kein Tippfehler, kleines o), der ein Paar von Punkten mit kleinster euklidischer Distanz unter allen Paaren bestimmt. Beweisen Sie, dass Ihr Algorithmus die geforderte Laufzeitschranke einhält.

43. Aufgabe

2 + 2 + 1 + 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 an, 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) Nehmen Sie an, die Wertigkeiten der k Münzen seien durch $1, b^1, b^2, \ldots, b^k$ für feste ganze Zahlen $b > 1$ und $k \geq 1$ gegeben. Beweisen Sie, dass das Münzwechselproblem für solche Münzsätze stets mit einem Greedy-Algorithmus gelöst werden kann.

c) 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.

d) Geben Sie einen Algorithmus an, der das Münzwechselproblem (exakt) löst. Sie dürfen für die Formulierung des Algorithmus ein festes, aber beliebiges M annehmen.

Bestimmen Sie die Laufzeit Ihres Algorithmus und begründen Sie Ihre Behauptungen. Ist Ihr Algorithmus ein Polynomialzeitalgorithmus (im Sinne von $\mathcal{P}$)?

44. Aufgabe

3 Punkte

Sei $x = x_1, \ldots, x_n$ eine Folge von natürlichen Zahlen. Wir nennen

$$x_{i_1}, \ldots, x_{i_k}$$

mit $i_j \in [1..n]$ für alle $j \in [1..k]$ eine *monotone Teilfolge* von x, wenn für alle $j \in [1..k-1]$ gilt, dass

$$i_j < i_{j+1} \quad \text{und} \quad x_{i_j} < x_{i_{j+1}} \; .$$

Entwerfen Sie einen Algorithmus, der in Zeit $\mathcal{O}(n \log n)$ zu einer (beliebig) gegebenen Folge x eine längste monotone Teilfolge (von x) bestimmt.

Begründen Sie die Korrektheit Ihres Algorithmus und dass er die Laufzeitschranke einhält.

45. Aufgabe

4 Punkte

Formulieren Sie das SSSPP für Digraphen als lineares Programm. Halten Sie hierfür den Digraph $G = (V, E, g)$ allgemein, entwerfen Sie also eine Reduktionsvorschrift, die Instanzen des SSSPP auf lineare Programme abbildet.

Begründen Sie die Korrektheit Ihres Entwurfs und illustrieren Sie ihn anhand des Digraphen aus Beispiel 4.1.

46. Aufgabe

3 Punkte

Beweisen Sie: Rucksack ist $\mathcal{NP}$-vollständig.