TU Wien
Institut für Logic and Computation
Forschungsbereich Algorithms and Complexity

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 7

PDF erstellt am: 14. Mai 2024

Deadline für dieses Übungsblatt ist **Montag, 10.6.2024, 20:00 Uhr**. Damit Sie für diese Übung Aufgaben anerkannt bekommen können, gehen Sie folgendermaßen vor:

1. Öffnen Sie den TUWEL-Kurs der Lehrveranstaltung *186.866 Algorithmen und Datenstrukturen (VU 5.5)* und navigieren Sie zum Abschnitt *Übungsblätter*.
2. Teilen Sie uns mit, welche Aufgaben Sie gelöst haben **und** welche gelösten Aufgaben Sie gegebenenfalls in der Übungseinheit präsentieren können. Gehen Sie dabei folgendermaßen vor:
   - Laden Sie Ihre Lösungen in einem einzigen PDF-Dokument in TUWEL hoch.
     Link *Hochladen Lösungen Übungsblatt 7*
     Button *Abgabe hinzufügen* bzw. *Abgabe bearbeiten*
     PDF-Datei mit Lösungen hochladen und *Änderungen sichern*.
   - Kreuzen Sie an, welche Aufgaben Sie gegebenenfalls in der Übung präsentieren können. Die Lösungen der angekreuzten Aufgaben müssen im hochgeladenen PDF enthalten sein.
     Link *Ankreuzen Übungsblatt 7*
     Aufgaben entsprechend anhaken und *Änderungen speichern*.

Bitte beachten Sie:

- Bis zur Deadline können Sie sowohl Ihr hochgeladenes PDF, als auch Ihre angekreuzten Aufgaben beliebig oft überschreiben. Sollte kurz vor der Deadline etwas schief gehen (Ausfall TUWEL, Internet, Scanner, etc.) und Sie die Endversion mit allen gelösten Aufgaben nicht mehr hochladen können, haben Sie zumindest Ihre Lösungen teilweise schon hochgeladen und angekreuzt. Nach der Deadline ist keine Veränderung mehr möglich. Die Deadline ist strikt – es werden ausnahmslos keine Nachabgabeversuche (z.B. per E-Mail) akzeptiert.
- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen und hochladen, oder aber auch gut leserliche Scans bzw. Fotos von handschriftlichen Ausarbeitungen hochladen (beachten Sie die maximale Dateigröße).
- Beachten Sie die Richtlinien für das An- und Aberkennen von Aufgaben. Details dazu finden Sie in den Folien der Vorbesprechung.

Aufgabe 1. Wir betrachten eine Funktion $f\colon \mathbb{N} \to \mathbb{Z}$, die wie folgt definiert ist:

$$f(n) := \begin{cases} 1 & \text{wenn } n \in \{1, 2\} \\ |n \cdot f(n-2)| - (f(n-1))^2 & \text{wenn } n > 2 \text{ und } n \text{ gerade ist und} \\ |n \cdot f(n-1)| - (f(n-2))^2 & \text{wenn } n > 2 \text{ und } n \text{ ungerade ist.} \end{cases}$$

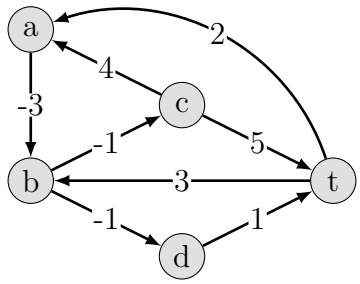
(a) Machen Sie sich mit der Funktion vertraut, indem Sie die Werte für $f(1), f(2), \ldots, f(8)$ angeben.

(b) Geben Sie einen naiven rekursiven Algorithmus (ohne dynamische Programmierung) an, der $f(n)$ berechnet.

(c) Adaptieren Sie Ihren naiven Algorithmus, sodass dieser dynamische Programmierung verwendet. Der Algorithmus soll immer noch rekursiv vorgehen. Geben Sie die Laufzeit des neuen Algorithmus in O-Notation an. Was ist der Vorteil gegenüber Ihrem Algorithmus aus Unteraufgabe (b)?

(d) Modifizieren Sie Ihren Algorithmus weiter, sodass dieser $f(n)$ iterativ und mittels dynamischer Programmierung arbeitet. Geben Sie die Laufzeit des neuen Algorithmus in O-Notation an.

Aufgabe 2. Gegeben ist folgender gerichteter Graph mit Kantengewichten.



(a) Wenden Sie den Algorithmus namens Simple-Shortest-Path-DP zur Berechnung von kürzesten Wegen aus der Vorlesung auf den Graphen oben an. Hierbei soll der Startknoten a sein und der Zielknoten t. Es genügt, den Tabelleninhalt nach jeder Iteration der Hauptschleife anzugeben.

Hinweis: Bevor Sie diese Teilaufgabe lösen, lesen Sie zuerst die zweite Teilaufgabe.

(b) Erklären Sie, wie Sie die Tabelle und den Algorithmus modifizieren können, sodass Sie einen negativen Kreis im Eingabegraphen finden können, sofern einer existiert. Nutzen Sie Ihre Einsicht, um mithilfe Ihrer Tabelle einen negativen Kreis im obigen Graphen zu finden.

Hinweis: Speichern Sie neben Zahlenwerten auch entsprechende Nachfolger in der Tabelle.

---

t						
a						
b						
c						
d						

Aufgabe 3. Gegeben ist folgende Instanz des Rucksackproblems:

#	Wert	Gewicht
1	2	1
2	5	2
3	8	3
4	10	4
5	12	5
6	14	6

Kapazität $G = 11$

(a) Wenden Sie den aus der Vorlesung bekannten Greedy-Algorithmus für das Rucksackproblem auf die obige Instanz an. Welche Gegenstände werden ausgewählt? Welchen Gesamtwert haben diese?

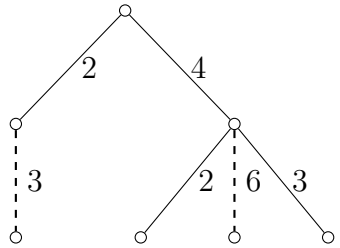
(b) Lösen Sie die obige Instanz jetzt durch dynamische Programmierung. Geben Sie dazu, wie aus der Vorlesung bekannt, die vollständige Belegung der 2-dimensionalen Lösungstabelle an.

Geben Sie die Menge der für die optimale Lösung ausgewählten Gegenstände an und markieren Sie in der Tabelle durch Einkreisen all jene Felder, die der Algorithmus `Find-Solution` aus der Vorlesung bei der Berechnung der Lösungsmenge A ausliest und verwendet. Welchen Gesamtwert hat die optimale Lösung?

	0	1	2	3	4	5	6	7	8	9	10	11
$\emptyset$												
$\{1\}$												
$\{1, 2\}$												
$\{1, 2, 3\}$												
$\{1, 2, 3, 4\}$												
$\{1, 2, 3, 4, 5\}$												
$\{1, 2, 3, 4, 5, 6\}$												

Aufgabe 4. Lösen Sie das folgende Problem mit dynamischer Programmierung. Als Eingabe bekommen Sie einen Wurzelbaum $T = (V, E)$ und für jede Kante $e \in E$ ein Kantengewicht $w_e \in \mathbb{R}$. Finden sollen Sie ein Matching $M \subseteq E$ in T mit maximalem Kantengewicht, d.h. $\sum_{e \in M} w_e$ soll maximal sein. Unten schreiben wir auch *Gewicht von* M statt $\sum_{e \in M} w_e$. Zur Vereinfachung sagen wir unten auch, dass ein Knoten v von einem Matching M *gematcht* wird, wenn eine mit v inzidente Kante in M enthalten ist.

Hier ist ein Beispielwurzelbaum zusammen mit einem Matching (gestrichelt) mit maximalem Kantengewicht:



Zum Lösen des Problems können Sie zwei Tabellen N und Y benutzen, die mit den Knoten von T indexiert sind. Für einen Knoten $v \in V$ bezeichnen wir dabei mit T_v denjenigen maximalen Teilbaum von T dessen Wurzel v ist. Für jeden Knoten $v \in V$ definieren wir die Tabellen wie folgt:

$$\begin{aligned} N[v] := \max\{w \mid &\text{ es existiert ein Matching } M \text{ im Teilbaum } T_v \\ &\text{mit Gewicht } w, \text{ sodass } v \textit{ nicht} \text{ von } M \text{ gematcht wird}\} \\ Y[v] := \max\{w \mid &\text{ es existiert ein Matching } M \text{ im Teilbaum } T_v \\ &\text{mit Gewicht } w, \text{ sodass } v \text{ von } M \text{ gematcht wird}\}. \end{aligned}$$

Dabei nehmen wir die Maxima als 0 an, wenn ein entsprechendes Matching die leere Menge ist oder nicht existiert.

Formulieren Sie einen Pseudocode, der mit dynamischem Programmieren die obigen Tabellen korrekt berechnet und das maximale Gewicht eines Matchings in T korrekt zurückgibt.

Hinweis: Lassen Sie sich von dem dynamischen Programm zu Weighted Independent Set auf Bäumen aus der Vorlesung inspirieren.

Aufgabe 5. Gegeben sei eine Folge $A = \langle a_1, a_2, a_3, \ldots, a_{n-1}, a_n \rangle$ von n natürlichen Zahlen und eine natürliche Zahl k. Ihre Aufgabe ist es, die Länge der längsten „k-chaotischen“ Teilfolge in A zu bestimmen: In einer solchen Teilfolge ist der Betrag der Differenz von je zwei aufeinanderfolgenden Zahlen mindestens k. Entwerfen und beschreiben Sie einen Algorithmus mittels dynamischer Programmierung, um dieses Problem zu lösen. Die Laufzeit soll $O(n^2)$ nicht überschreiten.

Beispiel: Sei $A = \langle 3, 4, 1, 2, 3 \rangle$, $k = 2$. Die maximalen zulässigen k-chaotischen Teilfolgen sind $\langle 3, 1, 3 \rangle$, $\langle 4, 1, 3 \rangle$, und $\langle 4, 2 \rangle$.

Anmerkung: Der Algorithmus muss nicht in Pseudocode angegeben werden. Es reicht, wenn die Funktionsweise klar und eindeutig beschrieben wird.
