TU Wien
Institut für Logic and Computation
Forschungsbereich Algorithms and Complexity

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 6

PDF erstellt am: 23. Mai 2022

Deadline für dieses Übungsblatt ist **Montag, 06.06.2022, 20:00 Uhr**. Um Aufgaben für diese Übung anerkannt zu bekommen, 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 6*
     Button *Abgabe hinzufügen*
     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 6*
     Button *Abgabe bearbeiten*
     Bearbeitete Aufgaben anhaken und *Änderungen speichern*.

Bitte beachten Sie:

- Bis zur Deadline können Sie sowohl Ihr hochgeladenes PDF, als auch Ihre angekreuzten Aufgaben beliebig oft verändern. Nach der Deadline ist keine Veränderung mehr möglich. Es werden ausnahmslos keine Nachabgabeversuche (z.B. per E-Mail) akzeptiert.
- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen, 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 finden Sie in den Folien der Vorbesprechung).

Aufgabe 1. Wenden Sie den Branch-and-Bound Algorithmus aus der Vorlesung auf die unten angegebene Instanz des Rucksackproblems an. Stellen Sie den Ablauf des Algorithmus als Baum dar. Geben Sie für jeden Schritt die obere Schranke U', die untere Schranke L', sowie eine passende Auswahl von Gegenständen mit Gesamtwert L' an.

Gegenstand	1	2	3	4	5
Gewicht g_i	20	32	15	12	40
Wert w_i	24	40	21	30	64

Rucksackkapazität: 100

Aufgabe 2. Entwerfen und beschreiben Sie einen Branch-and-Bound-Algorithmus für das Optimierungsproblem MAX-3SAT, das wie folgt definiert ist:

> Gegeben sei eine Menge $X = \{x_1, x_2, \ldots, x_n\}$ von Boole'schen Variablen sowie eine Menge $F = \{C_1, C_2, \ldots, C_m\}$ von Klauseln mit genau drei Literalen, d.h. für jede Klausel C_i gilt $C_i = (l_i^1 \vee l_i^2 \vee l_i^3)$, wobei $l_i^k \in \{x_j, \overline{x}_j\}$ für $1 \leq k \leq 3$ und $1 \leq j \leq n$. Wir nehmen außerdem an, dass in jeder Klausel drei verschiedene Variablen vorkommen und alle Klauseln (als Mengen von Literalen) unterschiedlich sind. Ziel ist eine Belegung der Variablen zu finden, die so viele Klauseln wie möglich erfüllt.

Gehen Sie in Ihrer Lösung insbesondere auf folgende Punkte ein (Pseudocode ist nicht notwendig):

(a) Wie repräsentieren Sie ein Teilproblem?

(b) Wie erfolgt das Branching, d.h. das Aufteilen eines (Teil-)Problems in weitere Teilprobleme? Beschreiben Sie eine sinnvolle Heuristik.

(c) Wie erstellen Sie eine gültige initiale Lösung? Zeigen Sie, dass immer eine Lösung gefunden werden kann, die mindestens die Hälfte der Klauseln in F erfüllt.

Aufgabe 3. Betrachten Sie erneut das in Aufgabe 2 definierte MAX-3SAT Problem.

(a) Zeigen Sie, dass das folgende Ja/Nein-Problem NP-schwer ist: Gegegeben sind eine Instanz von MAX-3SAT und eine natürliche Zahl k. Gibt es eine Belegung, die mindestens k Klauseln erfüllt?

(b) Betrachten Sie eine zufällig gewählte Wahrheitsbelegung, für die jede Variable x_j mit Wahrscheinlichkeit $p = 0.5$ auf wahr gesetzt wird. Wie viele Klauseln erfüllt eine so gewählte Belegung im Durchschnitt?

(c) Sei $C_i \in F$ eine Klausel. Wie viele Belegungen der Variablen X erfüllen diese Klausel *nicht*? Wie viele Klauseln m muss F *mindestens* enthalten, wenn *keine* erfüllende Belegung existiert?

Aufgabe 4. Betrachten Sie die folgende Funktion:

$$f(n) = \begin{cases} 1 & \text{wenn } n = 1, n = 2 \\ f(n-1) + f(n-2) & \text{wenn } n > 2 \text{ und } n \text{ ungerade} \\ 2 \cdot f(n-1) - f(n-2) & \text{wenn } n > 2 \text{ und } n \text{ gerade} \end{cases}$$

(a) Machen Sie sich mit der Funktion vertraut, indem Sie die Werte für $f(1), f(2), \ldots, f(9)$ angeben.

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

(c) Adaptieren Sie Ihren naiven Algorithmus, sodass dieser jetzt 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) Wandeln Sie Ihren Algorithmus erneut um, sodass dieser $f(n)$ jetzt iterativ und mittels dynamischer Programmierung berechnet. Geben Sie die Laufzeit des neuen Algorithmus in O-Notation an.

Aufgabe 5. Lösen Sie die folgende Instanz des gewichteten Interval Scheduling Problems mittels Dynamischer Programmierung wie aus der Vorlesung bekannt. Berechnen Sie für jeden Job j den Wert $p(j)$. Geben Sie das Array M mit allen Werten $M[j]$ an, sowie jeweils die benötigten Werte $w_j + M[p(j)]$ und $M[j-1]$. Berechnen Sie anschließend aus diesem Array eine Lösung und erklären Sie, wie Sie zu dieser Lösung gekommen sind.

Job	Start	Ende	Gewicht
1	1	10	2
2	8	12	6
3	8	15	7
4	13	20	4
5	14	30	3
6	22	31	5
7	11	35	5
8	21	40	8
9	37	42	4
10	32	50	3

---

j	1	2	3	4	5	6	7	8	9	10
$p(j)$										

j	1	2	3	4	5	6	7	8	9	10
$w_j + M[p(j)]$										
$M[j-1]$										
$M[j]$										

Aufgabe 6. Gegeben sei eine Folge $A = \langle a_1, a_2, a_3, \ldots, a_{n-1}, a_n \rangle$ von n natürlichen Zahlen. Ihre Aufgabe ist es, die Länge der längsten absteigenden Teilfolge in A zu bestimmen, in der je zwei aufeinanderfolgende Zahlen teilerfremd sind. Entwerfen und beschreiben Sie einen Algorithmus mittels Dynamischer Programmierung, um dieses Problem zu lösen. Die Laufzeit soll $O(n^2)$ nicht überschreiten. Sie können zur Vereinfachung annehmen, dass die Zahlenwerte beschränkt sind und der größte gemeinsame Teiler zweier Zahlen in A in $O(1)$ berechnet werden kann.

Beispiel: Sei $A = \langle 3, 12, 10, 9, 4, 11, 6, 5, 4, 3 \rangle$. Die Folge $\langle 10, 9, 5, 3 \rangle$ als Teilfolge von A wäre zulässig, da die Folge absteigend ist und keine zwei aufeinanderfolgenden Zahlen einen gemeinsamen Teiler haben. Die Folge $\langle 12, 10, 9, 5, 4, 3 \rangle$ ist nicht zulässig, da sie zwar absteigend ist, aber 12 und 10 beide durch 2 teilbar sind. Die längste Teilfolge von A, die absteigend ist und in der alle aufeinanderfolgenden Zahlen teilerfremd sind, ist in diesem Beispiel $\langle 12, 11, 6, 5, 4, 3 \rangle$.

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