TU Wien
Institut für Logic and Computation
Forschungsbereich Algorithms and Complexity

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 6

PDF erstellt am: 15. Mai 2023

Deadline für dieses Übungsblatt ist **Montag, 05.06.2023, 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:

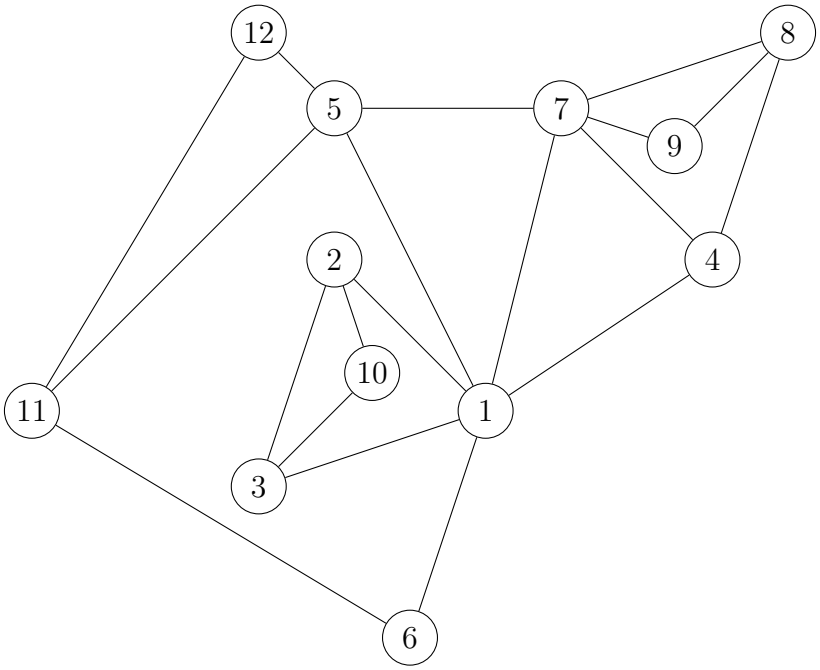
- Bis zur Deadline können Sie sowohl Ihr hochgeladenes PDF, als auch Ihre angekreuzten Aufgaben beliebig oft verändern. Nach der Deadline ist keine Veränderung mehr möglich. Es werden ausnahmslos keine Nachabgabeversuche (z.B. per E-Mail) akzeptiert.
- Sie können Ihre Lösungen entweder direkt in einem 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. Verwenden Sie die „Best-First“ Heuristik für die Auswahl von Teilproblemen.

Gegenstand	1	2	3	4	5
Gewicht g_i	30	20	50	10	45
Wert w_i	55	65	80	30	54

Rucksackkapazität: 100

Aufgabe 2. Wenden Sie den Branch-and-Bound Algorithmus zum Finden eines minimalen Vertex Covers auf den folgenden Graphen an. Geben Sie den Ablauf des Algorithmus als Baum wieder. Geben Sie für jeden Schritt die obere Schranke U' und die untere Schranke L' an, sowie die aktuelle Teilinstanz und die aktuelle Teillösung C'.



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

> Gegeben sei eine Menge von Boole'schen Variablen $V = \{x_1, x_2, \ldots, x_n\}$, sowie eine Menge $C = \{c_1, c_2, \ldots, c_m\}$ von Klauseln mit genau zwei Literalen, d.h. für jede Klausel c_i für $1 \leq i \leq m$ gilt $c_i = (l_i^1 \vee l_i^2)$, wobei $l_i^k \in \{x, \overline{x}\}$ für $k \in \{1, 2\}$ und ein $x \in V$. Ziel ist eine Belegung der Variablen zu finden, die so *wenige* Klauseln wie möglich erfüllt.

Gehen Sie in Ihrer Lösung insbesondere auf nachfolgende Punkte ein. Pseudocode ist nicht notwendig.

(a) Wie repräsentieren Sie ein Teilproblem und wie erfolgt das Branching, d.h. das Aufteilen eines (Teil-)Problems in weitere Teilprobleme?

(b) Beschreiben Sie eine sinnvolle Möglichkeit für die Auswahl des nächsten Teilproblems. Nach welcher Strategie gehen Sie dabei vor?

(c) Wie erstellen Sie eine gültige initiale Lösung? Kann die Lösung weniger als die Hälfte aller Klauseln in der Instanz erfüllen?

Aufgabe 4. Wir betrachten folgende Funktion $f\colon \mathbb{N} \to \mathbb{Z}$

$$f(n) = \begin{cases} 1 & \text{wenn } n = 1 \\ -1 & \text{wenn } n = 2 \\ |f(n-1)| + f(n-2) & \text{wenn } n > 2 \text{ und } f(n-1) \leq f(n-2) \\ f(n-2) - f(n-1) & \text{wenn } n > 2 \text{ und } f(n-1) > f(n-2) \end{cases}$$

(a) Machen Sie sich mit der Funktion vertraut, indem Sie die Werte für $f(1), f(2), \ldots, f(12)$ 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 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	12	3
2	10	20	2
3	14	23	2
4	8	30	1
5	21	34	4
6	24	40	5
7	19	44	3
8	14	45	2
9	41	51	2
10	50	60	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 ein Array A mit Länge n in dem jedes Element einen Wert aus $\{-1, 1, 2\}$ annimmt, es gilt also $A[i] \in \{-1, 1, 2\}$ für jedes $1 \leq i \leq n$. Betrachten Sie das Problem ein zusammenhängendes Teilarray mit mindestens einem Element zu finden, dessen Summe aller enthaltenen Werte maximal ist. Im Beispielarray

$$[1, -1, -1, \mathbf{2}, \mathbf{2}, \mathbf{-1}, \mathbf{-1}, \mathbf{2}, \mathbf{1}, -1]$$

wäre die Lösung das Teilarray vom vierten bis zum neunten Element mit Gesamtsumme 5.

Entwerfen Sie einen Algorithmus in Pseudocode, der den Wert des besten Teilarrays mithilfe dynamischer Programmierung und Speicherung der Zwischenlösungen in Arrays bestimmt und dafür Laufzeit $O(n)$ benötigt. Beschreiben Sie, wie man aus den gespeicherten Zwischenlösungen und der Lösung das optimale Teilarray finden kann.
