Algorithmen und Programmierung 3, WS 2019/2020 — 14. Übungsblatt

Abgabe bis Freitag, 7. Februar 2020, 12:00 Uhr, in die Fächer der Tutoren

---

92. Adjazenzlisten, 6 Punkte

    Gegeben ist ein Graph mit den Knoten $V = \{1, 2, \ldots, n\}$ und m Kanten in Adjazenzlistenspeicherung. Die Kanten (i, j) in den Adjazenzlisten sollen nach den Endknoten umsortiert werden; das heißt, in der Adjazenzliste von i soll die Kante (i, j) vor (i, k) kommen, wenn $j < k$ ist.

    Skizzieren Sie einen Algorithmus, der diese Sortieraufgabe *in linearer Zeit*, das heißt, in $O(m+n)$ Zeit löst. (Lassen Sie sich von Sortieren durch Fachverteilung/Bucketsort oder Radix-Sort mit Basis n inspirieren.)

93. Optimaler binärer Suchbaum, 7 Punkte

    Gegeben sind fünf Schlüssel $(x_1, x_2, x_3, x_4, x_5) = (12, 14, 17, 21, 23)$ mit den Anfragehäufigkeiten

    $$(p_1, p_2, p_3, p_4, p_5) = (4, 1, 6, 7, 2)$$

    Dabei ist p_i die Häufigkeit, mit der der Schlüssel x_i angefragt wird. Es werden nur vorhandene Schlüssel gesucht. (Die Größen $q_0, \ldots, q_5$ aus der Vorlesung sind 0.)

    Bestimmen Sie den optimalen binären Suchbaum mit dem dynamischen Programmierungsalgorithmus aus der Vorlesung, und berechnen Sie seine mittlere Suchzeit: den *Erwartungswert* der Anzahl der Schlüssel, mit denen der Suchwert x verglichen wird. Vergessen Sie nicht, das Ergebnis durch $p_1 + \cdots + p_5$ zu dividieren. Geben Sie auch die Zwischenergebnisse im Algorithmus an und erklären Sie, was sie bedeuten.

94. Schlechtester binärer Suchbaum, 5 freiwillige Zusatzpunkte als Bonus

    (a) Bestimmen Sie den Suchbaum für die Daten von Aufgabe 93 mit der *größten* mittleren Suchzeit.

    (b) Man kann zeigen, dass ein solcher Baum immer ein *entarteter Baum* sein muss, der auf jeder Ebene nur einen inneren Knoten hat. Entwerfen Sie mit Hilfe dieser Eigenschaft einen Algorithmus, der den schlechtesten Suchbaum in $O(n^2)$ Zeit bestimmt.

95. Optimaler binärer Suchbaum, Beschleunigung, 0 Punkte

    Wie bezeichnen mit k_{ij} den Index k, bei dem in der Rekursion

    $$S_{ij} = \min\{\, S_{ik} + S_{kj} \mid 1 < k < j \,\} + (q_i + p_{i+1} + \cdots + q_{j-1}) \qquad (*)$$

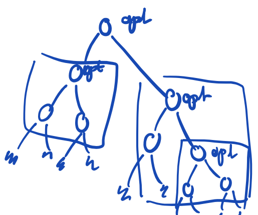
    das Minimum erreicht wird, für $0 \le i < j - 1 \le n$. Der Knoten x_k ist dann die Wurzel des optimalen Teilsuchbaumes. Falls es mehrere Minima gibt, wählen wir das kleinste k.

    (a) Man[1] kann zeigen, dass $k_{i,j-1} \le k_{ij} \le k_{i+1,j}$ ist, für $j \ge i + 3$. Wenn man diese Bedingung berücksichtigt, wie viele Möglichkeiten für k muss man dann in der ℓ-ten Diagonalen $(j - i = \ell)$ der Tabelle insgesamt ausprobieren?

    (b) Wie kann man durch diese Beobachtung die Rekursion $(*)$ in $O(n^2)$ Zeit lösen?

---

[1]D. E. Knuth, Optimum binary search trees, *Acta Informatica* **1** (1971), 14–25, doi:10.1007/BF00264289



96. Zahlen mit Münzen, 0 Punkte

An der Supermarktkasse soll ein Betrag von n Cents nur mit Münzen bezahlt werden, wobei die minimale Anzahl an Münzen verwendet werden soll. (Zur Erinnerung: Es gibt Euro-Münzen zu 1, 2, 5, 10, 20, 50, 100 und 200 Cent.)

(a) Beschreiben Sie einen gierigen Algorithmus für das Problem und implementieren Sie ihn. Was ist die Laufzeit des Algorithmus?

(b) Beweisen Sie, dass Ihr Algorithmus aus (a) eine optimale Lösung liefert.
Hinweis: Machen Sie sich Gedanken über die Struktur einer optimalen Lösung: Wie viele 1-Cent-Münzen kann eine optimale Lösung höchstens enthalten? Wie viele 2-Cent-Münzen? usw?

(c) Geben Sie ein Beispiel von Münzwerten, wo der gierige Algorithmus nicht optimal ist. Es soll eine 1-Cent-Münze geben, damit sich jeder Betrag zahlen lässt.

(d) Wie wäre es, wenn man wie in der USA die 20-Cent-Münze durch eine Viertel-Euro-Münze ersetzen würde?

(e) Beweisen Sie, dass der gierige Algorithmus für Münzen mit Werten $1, B, B^2, \ldots, B^k$ die minimale Anzahl von Münzen verwendet, wenn $B, k \in \mathbb{N}$ und $B > 1$ ist.

97. Zahlen mit Münzen, 7 Punkte

Entwerfen Sie einen Algorithmus zur optimalen Lösung des Münzwechselproblems nach dem Prinzip der dynamischen Programmierung. Die Eingabe ist die Folge $M_1 = 1 < M_2 < \cdots < M_n$ der ganzzahligen Münzwerte und der zu zahlende Betrag B.

Definieren Sie zuerst die *Teilprobleme*, und stellen Sie die Rekursionsgleichungen mit den Randbedingungen auf. Schreiben Sie *danach* den Algorithmus in Pseudocode.

98. Bezahlen mit Wechselgeld, 0 Punkte
Wie ist es, wenn man einen größeren Betrag zahlen darf und sich Wechselgeld herausgeben lässt? Die Zahl der ausgetauschten Münzen soll minimiert werden. Zum Beispiel ist $99 = 100 - 1$, das sind 2 Münzen. Gibt es einen einfachen (gierigen?) Algorithmus für die Euromünzen? Kann man dynamische Programmierung anwenden?

99. Fehlstände, 5 freiwillige Zusatzpunkte als Bonus

Ein *Fehlstand* oder eine *Inversion* in einer Folge $(x_1, \ldots, x_n)$ von Zahlen ist ein Paar (i, j) mit $1 \leq i < j \leq n$ und $x_i > x_j$. Die Anzahl der Fehlstände gibt an, wie viele benachbarte Vertauschungen bei Bubblesort durchgeführt werden. Die Folge $(2, 4, 6, 3, 5)$ hat zum Beispiel 3 Fehlstände.

Entwerfen Sie einen Algorithmus, der die Anzahl der Fehlstände in $O(n \log n)$ Zeit berechnet. Sie dürfen dabei geeignete Datenstrukturen zu Hilfe nehmen. Formulieren Sie den Algorithmus in Pseudocode, und erläutern Sie die Korrektheit.

100. Optimale Matrizenkettenmultiplikation, dynamisches Programmieren, 0 Punkte

Das Produkt von n Matrizen $M_1 M_2 \ldots M_n$ soll berechnet werden, wobei die i-te Matrix a_{i-1} Zeilen und a_i Spalten hat. Das Produkt kann von links nach rechts als $((M_1 \cdot M_2) \cdot M_3) \cdot M_4$ (für $n = 4$) oder von rechts nach links oder zum Beispiel als $M_1 \cdot ((M_2 \cdot M_3) \cdot M_4)$ oder auf noch viele andere Arten ausgerechnet werden.

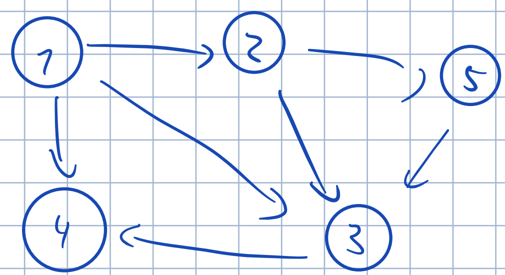
(a) Bestimmen Sie für $(a_0, a_1, a_2, a_3, a_4) = (40, 2, 10, 4, 8)$ für jede dieser drei Möglichkeiten die Gesamtanzahl an Einzelmultiplikationen zur Berechnung der (40×8)-Produktmatrix $M_1 M_2 M_3 M_4$. Das Produkt einer $(a \times b)$-Matrix mit einer $(b \times c)$-Matrix kann dabei mit abc Einzelmultiplikationen ausgerechnet werden.

(b) Entwerfen Sie einen effizienten Algorithmus, der die optimale Berechnungsmethode für allgemeines n und beliebige Abmessungen $(a_0, \ldots, a_n)$ bestimmt.

92. Adjazenzlisten, 6 Punkte   Von Thore Brehmer und David Ley

Gegeben ist ein Graph mit den Knoten $V = \{1, 2, \ldots, n\}$ und m Kanten in Adjazenzlistenspeicherung. Die Kanten (i, j) in den Adjazenzlisten sollen nach den Endknoten umsortiert werden; das heißt, in der Adjazenzliste von i soll die Kante (i, j) vor (i, k) kommen, wenn $j < k$ ist.

Skizzieren Sie einen Algorithmus, der diese Sortieraufgabe *in linearer Zeit*, das heißt, in $O(m+n)$ Zeit löst. (Lassen Sie sich von Sortieren durch Fachverteilung/Bucketsort oder Radix-Sort mit Basis n inspirieren.)



Vorbereitung:

Speicher der Adjazenzliste in zwei Listen. Eine Kantenliste der Länge m und eine Pointer/Knoten Liste der Länge n

Index	(0	1	2	3	4)
Knoten	1	2	3	4	5
Pointer:	[0,	3,	5,	/,	5]

Index	0	1	2	3	4	5	6
Kanten:	[3,	2,	4,	3,	5,	4,	3]

1. Addiere zu jeden Index der Kantenliste, abhängig von der Pointerliste, $10^{\left\lceil \frac{\text{Index von Pointerliste}}{10} \right\rceil}$ $O(n)$

Index	(0	1	2	3	4)
Knoten	1	2	3	4	5
$\Rightarrow$	[0,	3,	5,	/,	5]

Index	0	1	2	3	4	5	6
Kantenliste	[13,	12,	14,	23,	25,	34,	43]

2. Nun Sortiere die Kantenliste mit Bucketsort $O(n)$

$$\Rightarrow \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \end{array}$$
$$\left[12, 13, 14, 23, 25, 34, 43\right]$$

3. Mach den 1sten Schritt rückgängig. (Subtrahiere $10^{\left\lceil \frac{\text{Index von Pointerliste}}{10} \right\rceil}$) $O(n)$

$$\Rightarrow \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \end{array}$$
$$\left[2, 3, 4, 3, 5, 4, 3\right]$$

Die Adjazenzliste ist nun Sortiert. Dauer $O(3n) \in O(n) \Rightarrow$ linear

Index:	$(0\ 1\ 2\ 3\ 4)$
Knoten:	$1\ 2\ 3\ 4\ 5$
Pointer:	$[0, 3, 5, /, 5]$

Index:	$0\ 1\ 2\ 3\ 4\ 5\ 6$
Kantenliste:	$[2, 3, 4, 3, 5, 4, 3]$

93. Optimaler binärer Suchbaum, 7 Punkte

Gegeben sind fünf Schlüssel $(x_1, x_2, x_3, x_4, x_5) = (12, 14, 17, 21, 23)$ mit den Anfragehäufigkeiten

$$(p_1, p_2, p_3, p_4, p_5) = (4, 1, 6, 7, 2)$$

Dabei ist p_i die Häufigkeit, mit der der Schlüssel x_i angefragt wird. Es werden nur vorhandene Schlüssel gesucht. (Die Größen $q_0, \ldots, q_5$ aus der Vorlesung sind 0.)

Bestimmen Sie den optimalen binären Suchbaum mit dem dynamischen Programmierungsalgorithmus aus der Vorlesung, und berechnen Sie seine mittlere Suchzeit: den *Erwartungswert* der Anzahl der Schlüssel, mit denen der Suchwert x verglichen wird. Vergessen Sie nicht, das Ergebnis durch $p_1 + \cdots + p_5$ zu dividieren. Geben Sie auch die Zwischenergebnisse im Algorithmus an und erklären Sie, was sie bedeuten.

w := weight
r := root
c := cost

$w_{ij} = w_{ij-1} + q_j + p_j$
$c_{ij} = w_{ij} + \min(c_{im-1} + c_{mj}) \quad i < m \le j$
$r_{ij} = a_m$

4, 1, 6, 7, 2
1 2 3 4 5

$c_{0,2} = 5 + \min(\underset{0}{c_{0,0}} + \underset{1}{c_{1,2}}, \underset{4}{c_{0,1}} + \underset{0}{c_{22}}) \Rightarrow c = 6, r = 1$

$c_{1,3} = 7 + \min(\underset{0}{c_{1,1}} + \underset{6}{c_{2,3}}, \underset{1}{c_{1,2}} + \underset{0}{c_{33}}) \Rightarrow c = 8, r = 3$

$c_{2,4} = 13 + \min(\underset{0}{c_{22}} + \underset{7}{c_{3,4}}, \underset{6}{c_{2,3}} + \underset{0}{c_{44}}) \Rightarrow c = 13, r = 4$

$c_{3,5} = 9 + \min(\underset{0}{c_{33}} + \underset{2}{c_{4,5}}, \underset{7}{c_{3,4}} + \underset{0}{c_{55}}) \Rightarrow c = 11, r = 4$

$c_{0,3} = 11 + \min(8, 12, 6) \Rightarrow c = 17, r = 3$

$c_{1,4} = 14 + \min(19, 8, 8) \Rightarrow c = 22, r = 3$

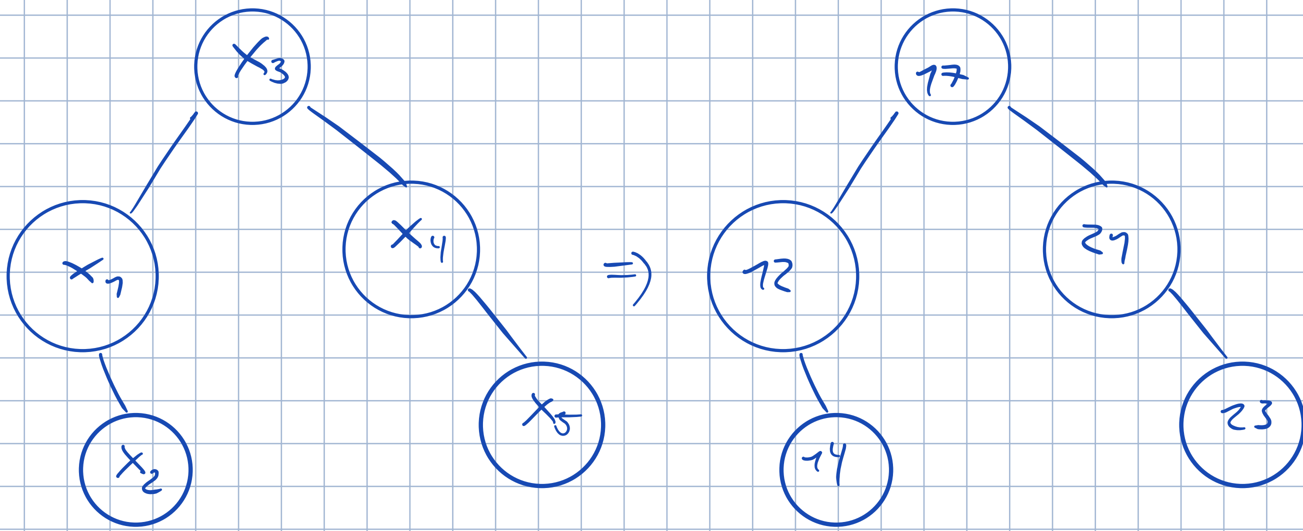
$c_{2,5} = 15 + \min(11, 8, 19) \Rightarrow c = 23, r = 4$

$c_{0,4} = 18 + \min(22, 23, 13, 17) \Rightarrow c = 31, r = 3$

$c_{1,5} = 16 + \min(23, 15, 8, 17) \Rightarrow c = 24, r = 4$

$c_{0,5} = 20 + \min(24, 27, 18, 19, 31) \Rightarrow c = 38, r = 3$

0	1	2	3	4	5	j/i
w = 0 c = 0 r = ∅	w = 4 c = 4 r = 1	w = 5 c = 6 r = 1	w = 11 c = 17 r = 3	w = 18 c = 31 r = 3	w = 20 c = 37 r = 3	0
	w = 0 c = 0 r = ∅	w = 1 c = 1 r = 2	w = 7 c = 8 r = 3	w = 14 c = 22 r = 3	w = 16 c = 26 r = 4	1
		w = 0 c = 0 r = ∅	w = 6 c = 6 r = 3	w = 13 c = 19 r = 4	w = 15 c = 23 r = 4	2
			w = 0 c = 0 r = ∅	w = 7 c = 7 r = 4	w = 9 c = 11 r = 4	3
				w = 0 c = 0 r = ∅	w = 2 c = 2 r = 5	4
					w = 0 c = 0 r = ∅	5



$$\text{Erwartungswert} = \frac{38}{p_1 \ldots p_5} = \frac{38}{20} = 1{,}9$$