



## Effiziente Algorithmen SS 2009 Übungsblatt 7

Besprechung: 28. und 29. Kalenderwoche (2009)

### Aufgabe 1 (Huffman-Codes)

Wenden Sie den Huffman-Algorithmus zur Berechnung der impliziten Darstellung eines Huffman-Baumes sowie den Algorithmus zum expliziten Aufbau des Huffman-Baumes aus dieser Darstellung auf das folgende Beispiel an.

$a$	A	B	C	D	E	F	G	H	I	K
$p(a)$	0.21	0.07	0.15	0.10	0.09	0.08	0.07	0.08	0.09	0.06

Berechnen Sie  $B(T, p)$  für Ihren Baum.

### Aufgabe 2 (Union-Find-Datenstrukturen)

Betrachten Sie die beiden in der Vorlesung erläuterten Implementierungen einer Union-Find-Datenstruktur.

- Arrays und Listen: Finden Sie eine Sequenz aus  $\text{INIT}(n)$ , gefolgt von  $n - 1$  UNION-Operationen, die  $\Omega(n \log n)$  Zeit beansprucht.
- Wurzelgerichtete Wälder ohne Pfadkompression: Geben Sie eine Sequenz aus  $\text{INIT}(n)$ , gefolgt von  $n - 1$  UNION-Operationen an, sodass danach die Sequenz der Operationen  $\text{FIND}(1), \dots, \text{FIND}(n)$  insgesamt  $\Omega(n \log n)$  Zeit beansprucht.
- Wurzelgerichtete Wälder mit Pfadkompression: Zeigen Sie, dass die Ränge entlang eines wurzelgerichteten Weges strikt anwachsen (Fakt 1).

### Aufgabe 3 (Zusammenhangskomponenten)

Sei  $G = (V, E)$ ,  $V = \{1, \dots, n\}$ , ein ungerichteter Graph, der als ungeordnete Kantenliste  $(\{v_1, w_1\}, \dots, \{v_m, w_m\})$  gegeben ist. Wie und mit welcher Laufzeit könnte man ohne Tiefensuche die Zusammenhangskomponenten finden?

### Aufgabe 4 (APSP-Algorithmus von Floyd-Warshall)

- Der Algorithmus von Floyd-Warshall liefert mit der Ausgabe von  $I$  eine implizite Darstellung kürzester Wege zwischen je zwei Knoten  $v$  und  $w$  des Eingabe-Graphen  $G$ . Formulieren Sie eine (rekursive) Prozedur  $\text{PRINTPATH}(v, w)$ , die einen kürzesten Weg von  $v$  nach  $w$  als Knotenfolge ausgibt und dazu nur  $I$  benutzt. Beweisen Sie die Korrektheit Ihrer Prozedur.
- Angenommen, man lässt für Eingabe-Graphen  $G$  auch negative Kantengewichte zu. Wie muss man den Algorithmus von Floyd-Warshall modifizieren, sodass er feststellt, ob in  $G$  ein Kreis mit negativer Länge vorhanden ist?

**Aufgabe 5** (Rucksackproblem)

Betrachten Sie den nach dem Prinzip der dynamischen Programmierung gestalteten Algorithmus zur Lösung des Rucksackproblems. Davon gibt es eine Variante, die bei Eingabe  $(a_1, \dots, a_n, c_1, \dots, c_n, b)$  Werte  $M(i, j)$  für  $i = 1, \dots, n$  und  $j = 0, 1, \dots, \hat{c}$  berechnet mit  $M(i, j) :=$  minimales Volumen einer legalen Bepackung mit Objekten der Menge  $\{1, \dots, i\}$  und Nutzen  $j$ , falls eine solche existiert. Dabei  $\hat{c} := c_1 + \dots + c_n$ .

- Stellen Sie die Bellmanschen Optimalitätsgleichungen für diese Variante auf und begründen Sie die Gleichungen möglichst genau.
- Formulieren Sie den Algorithmus zur Berechnung der  $M(i, j)$ -Werte und eventuell von Hilfsinformation (für (c)).
- Geben Sie eine Prozedur zur Konstruktion einer optimalen Bepackung an.

**Aufgabe 6** (Reklametafeln am Straßenrand)

Angenommen, Sie sind für das Aufstellen von Reklametafeln entlang der A3 Richtung Frankfurt verantwortlich, einem stark befahrenen Streckenabschnitt von  $M$  Kilometern. Die möglichen Plätze für Reklametafeln sind durch  $n$  aufsteigend sortierte natürliche Zahlen  $x_1, \dots, x_n$  im Intervall  $[0, M]$  gegeben, die die Positionen entlang der Autobahn jeweils als Entfernung in Kilometern vom nördlichen Ende des Streckenabschnitts spezifizieren. Wenn Sie eine Reklametafel an Position  $x_i$  anbringen, bekommen Sie einen Erlös von  $r_i > 0$  Euro.

Das Straßenbauamt hat festgelegt, dass je zwei dieser Reklametafeln einen Abstand von mehr als 5 Kilometern voneinander haben müssen. Sie würden die Reklametafeln nun gerne so aufstellen, dass unter Einhaltung der Verordnung des Straßenbauamtes der Gesamterlös maximiert wird.

Geben Sie einen Algorithmus an, der in Zeit  $O(n)$  zu einer Instanz dieses Problems den mit einer zulässigen Platzierung der Reklametafeln maximal erzielbaren Erlös berechnet (und natürlich eine Platzierung, die dies realisiert!).

**Aufgabe 7** (Traveling Salesman Problem)

Das Traveling Salesman Problem, kurz TSP, ist folgendermaßen definiert: Gegeben  $n$  natürlich durchnummerierte Städte und eine  $n \times n$ -Matrix  $(c_{i,j})$ , sodass Eintrag  $c_{i,j}$  die Kosten einer direkten Verbindung zwischen den Städten  $i$  und  $j$  repräsentiert, finde eine billigste Rundreise, d. h., eine Permutation  $\pi \in S_n$ , die die Summe

$$\sum_{i=1}^{n-1} c_{\pi(i), \pi(i+1)} + c_{\pi(n), \pi(1)}$$

minimiert. Falls keine direkte Verbindung von  $i$  nach  $j$  existiert, ist  $c_{i,j} = \infty$ .

Für  $S \subseteq \{1, \dots, n\}$ ,  $1, j \in S$  und  $j \neq 1$ , sei der Wert  $l(S, j)$  definiert als die Länge eines kürzesten Weges, der in 1 startet und in  $j$  endet und jede Stadt in  $S$  genau einmal besucht.

- Zeigen Sie, wie man iterativ über  $s = |S|$  in Zeit  $O(2^n \cdot n^2)$  alle Werte  $l(S, j)$  berechnen kann.
- Zeigen Sie, wie man dann eine Lösung für das TSP in Zeit  $O(n)$  konstruieren kann.

**Bemerkung:** Der naive Ansatz zur Lösung des TSPs hat eine Laufzeit von  $\Omega(n!)$ , was wesentlich größer als  $O(2^n \cdot n^2)$  ist.

**Aufgabe 8** (Besondere Aufgabe auf English: bis 16.07.2009)

We are looking at the price of a given stock over  $n$  consecutive days, numbered  $i = 1, 2, \dots, n$ . For each day  $i$ , we have a price  $p(i)$  per share for the stock on that day. ( We assume for simplicity that the price was fixed during each day.) We would like to know: How should we choose a day  $i$  on which to buy the stock and a later day  $j > i$  on which to sell it, if we want to maximize the profit per share  $p(j) - p(i)$ ? ( If there is no way to make money during the  $n$  days, we should conclude this instead.)

Show how to find the optimal numbers  $i$  and  $j$  in time  $O(n)$ .

**Aufgabe 9** (Besondere Aufgabe auf English: bis 16.07.2009)

Show that an unweighted graph with  $n$  nodes has at most  $n(n - 1)$  distinct minimum cuts. (A cut is a set of edges whose removal leaves a graph disconnected.)

**Aufgabe 10** (Besondere Aufgabe auf English: bis 16.07.2009)

Give a randomized algorithm computing a minimum cut with high probability in time  $O(mn^2 \log n)$ . (A cut is a set of edges whose removal leaves a graph disconnected.)

**Aufgabe 11** (Besondere Aufgabe auf English: bis 16.07.2009)

To assess how “well-connected two nodes in a directed graph are, one may not only look at the length of the shortest path between them, but may also count the number of shortest paths.

This turns out to be a problem that can be solved efficiently, subject to some restriction on edge costs. Suppose we are given a directed graph  $G = (V, E)$  with costs on the edges; the costs may be positive or negative, but every cycle in the graph has strictly positive cost. We are also given two nodes  $v, w \in V$ . Give an efficient algorithm that computes the number of shortest  $v - w$  paths in  $G$ . (The algorithm should not list all the paths; just the number suffices.

**Aufgabe 12** (Besondere Aufgabe auf English: bis 16.07.2009)

Lucia wants to throw a party and is deciding whom to call. She has  $n$  people to choose from, and she has made up a list of which pairs of these people know each other. She wants to foster an environment where people can feel both comfortable with old buddies and overwhelmed with new surprises. Hence, she wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least  $k$  people whom they know and  $k$  people whom they don't know, for a constant  $k \geq 1$ .

Give an efficient algorithm that takes as input the list of  $n$  people and the list of pairs who know each other and outputs the best choice of party invitees. Give the running time in terms of  $n$ .