Entwurf und Analyse von Algorithmen

WS 1999/2000
Prof. Dr. K. Madlener
Übungsblatt 8

Aufgabe 8.1. a) Konstruieren Sie den Pattern-Matching-Automaten für das Muster `aabab` und illustrieren Sie seine Arbeitsweise für den Text `aaababaabaababaab`.

b) Berechnen Sie die Verschiebetabelle, die im Knuth-Morris-Pratt-Algorithmus benutzt wird, für das Muster `ababbabbababbababbabb` und das Alphabet $\{\mathtt{a}, \mathtt{b}\}$.

c) Gegeben seien zwei Wörter x und y. Geben Sie einen Algorithmus an, der entscheidet, ob x und y durch zyklische Rotation auseinander hervorgehen. Beispiel: Auf `arc` und `car` trifft dies zu, nicht aber auf `regal` und `lager`. Ihr Algorithmus soll in Zeit $O(|x| + |y|)$ laufen.

Aufgabe 8.2. In dieser Aufgabe beschäftigen wir uns mit (Anwendungen von) Suffixbäumen.

a) Zeichnen Sie den Suffixbaum für das Wort `aabbabd`.

b) Es seien k Wörter gegeben. Gesucht ist das längste gemeinsame Teilwort aller gegebenen Wörter. Geben Sie eine Lösung an, die dieses längste gemeinsame Teilwort in Zeit $O(n)$ berechnet, wobei n die Gesamtlänge aller k Wörter sei. (Dabei sollen k sowie die Alphabetgröße als konstant angesehen werden.)

Hinweis: Konstruieren Sie (im Falle $k = 2$) für gegebene Wörter x und y einen Suffixbaum für das Wort $x\#y$ (für einen „neuen“ Buchstaben $\#$) und fügen Sie in geeigneter Reihenfolge allen Knoten im Suffixbaum Informationen hinzu, so dass eine anschließende Traversierung der Knoten das gewünschte Ergebnis liefert.

c) Es sei ein Wort x gegeben. Gesucht ist das längste Teilwort, das mehr als einmal vorkommt (wenn es ein solches gibt). Geben Sie eine Lösung an, die dieses längste wiederholte Teilwort in Zeit $O(|x|)$ berechnet.

Aufgabe 8.3. Wir erinnern uns: Eine Clique in einem ungerichteten Graphen $G = (V, E)$ ist eine Knotenteilmenge $S \subseteq V$, so dass $\{x, y\} \in E$ für alle $x, y \in S$. Außerdem wollen wir $\emptyset$ sowie $\{x\}$ für alle $x \in V$ als Cliquen ansehen. Eine Clique heiße maximal, wenn sie keine Teilmenge einer anderen Clique ist.

Wir wollen folgendes Problem betrachten: Es sei ein ungerichteter Graph $G = (V, E)$ gegeben, und es sollen alle Cliquen von G ohne Wiederholung aufgezählt werden.

Zur Lösung dieses Problems sei bemerkt, dass offensichtlich eine Folge $(x_0, x_1, \ldots, x_{\ell-1})$ genau dann eine partielle Lösung ist, wenn $\{x_0, x_1, \ldots, x_{\ell-1}\}$ eine Clique ist (zeigen Sie dies). Im Folgenden bezeichne C_ℓ die Menge der Knoten, so dass $(x_0, x_1, \ldots, x_\ell)$ eine partielle Lösung ist, falls $(x_0, x_1, \ldots, x_{\ell-1})$ eine partielle Lösung ist. Wenn wir nun $S_{\ell-1} = \{x_0, \ldots, x_{\ell-1}\}$ und $C_0 = V$ setzen, dann ist C_ℓ gegeben durch

$$\begin{aligned} C_\ell &= \{v \in V \setminus S_{\ell-1} : \{v, x\} \in E \text{ für alle } x \in S_{\ell-1}\} \\ &= \{v \in C_{\ell-1} \setminus \{x_{\ell-1}\} : \{v, x_{\ell-1}\} \in E\}. \end{aligned}$$

Zeigen Sie dies.

Leider produziert ein Backtracking-Algorithmus, der eine solche Strategie verwendet, eine jede Clique der Größe k genau $k!$-mal (zeigen!). Um diese Mehrfacharbeit zu vermeiden, ordnen wir die Knoten in V willkürlich durch eine totale Ordnung, d. h. wir stellen uns V als eine geordnete Liste vor: $V = (v_0, v_1, \ldots, v_{n-1})$ mit $v_0 < v_1 < \cdots < v_{n-1}$. Zeigen Sie, dass folgende Neudefinition von C_ℓ sinnvoll ist:

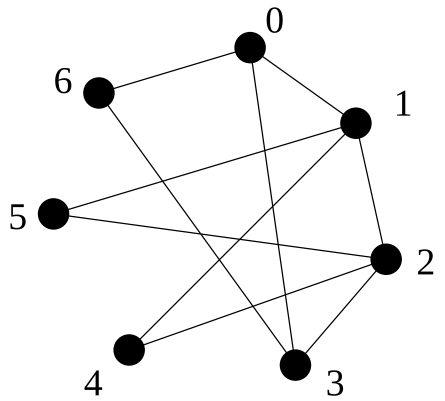
$$C_\ell = \{v \in C_{\ell-1} : \{v, x_{\ell-1}\} \in E \text{ und } v > x_{\ell-1}\}.$$

Man kann C_ℓ effizienter berechnen, wenn man für jeden Knoten $v \in V$ die Hilfsmengen $A_v = \{u \in V : \{u, v\} \in E\}$ und $B_v = \{u \in V : u > v\}$ bestimmt (als Vorberechnungs-Schritt). Zeigen Sie, dass dann $C_\ell = A_{x_{\ell-1}} \cap B_{x_{\ell-1}} \cap C_{\ell-1}$ ist.

Setzt man nun noch $N_\ell = N_{\ell-1} \cap A_{x_{\ell-1}}$ und $N_0 = V$, dann ist $(x_0, \ldots, x_{\ell-1})$ genau dann eine maximale Clique, wenn $N_\ell = \emptyset$ ist (zeigen!).

Geben Sie nun einen Backtracking-Algorithmus an, der die angedeutete Strategie benutzt. Er soll jede Clique genau einmal aufzählen und maximale Cliquen sogleich als solche identifizieren.

Zeigen Sie die Arbeitsweise Ihres Algorithmus für folgenden Graphen:



Geben Sie für alle Knoten v die Mengen A_v und B_v an und zeichnen Sie den Lösungsbaum, den Ihr Algorithmus generiert. Markieren Sie dort Knoten, die maximalen Cliquen entsprechen.

Sei nun n eine positive ganze Zahl und sei weiter $\mathcal{G}(n)$ die Menge aller ungerichteten Graphen mit Knotenmenge $V_n = \{0, 1, \ldots, n-1\}$. Zeigen Sie, dass $\mathcal{G}(n)$ genau $2^{\binom{n}{2}}$ Graphen enthält.

Für einen Graphen $G \in \mathcal{G}(n)$ bezeichne $c(G)$ die Anzahl der Cliquen in G. Hat Ihr Algorithmus eine Laufzeit in $O(n\ c(G))$? Welche Laufzeit hat Ihr Algorithmus für den Graphen, der keine Kanten hat (und Knotenmenge V_n)? Für den vollständigen Graphen K_n auf V_n?

Optionale Zusatzaufgabe: Versuchen Sie, die mittlere Laufzeit Ihres Algorithmus zu ermitteln (unter der Annahme, dass alle Graphen in $\mathcal{G}(n)$ gleichwahrscheinlich als Eingabe auftreten).

Bemerkung: Es gibt einen Algorithmus, der das oben angedeutete Verfahren benutzt und dessen mittlere Laufzeit in $O(n^{\log_2 n + 2})$ liegt.

Informationen zur Vorlesung:
http://www-madlener.informatik.uni-kl.de/ag-madlener/lectures/eaa/ws1999-2000/eaa.html