Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr

WS 2000/2001
Übungsblatt 12
16. Januar 2001

Diskrete Strukturen I

Abgabetermin für die Computeraufgaben: Dienstag, den 23. Januar 2001 (16.^{00} Uhr)
(Aufgabensteller: Ulrich Voll)

Tutoraufgabe 1

a) In einem Baum T auf der n-elementigen Knotenmenge V gibt es immer einen Knoten v^*, so daß der Wald induziert von $V \setminus v^*$ aus Bäumen besteht, die jeweils maximal $\lfloor n/2 \rfloor$ Knoten haben.

b) Sei $G = (V, E)$ ein beliebiger Graph. Eine Kante $e \in E$ ist genau dann Brücke, wenn sie auf keinem Kreis liegt. (*Erinnerung:* Eine Kante ist genau dann Brücke, wenn ihre Entfernung die Anzahl der Zusammenhangskomponenten erhöht.)

c) Sei nun G derart, daß alle Knoten geraden Grad haben. Zeigen Sie, daß ein solcher Graph keine Brücke haben kann.

d) Sei T ein gewurzelter Binärbaum mit n Knoten (jeder innere Knoten hat maximal zwei Kinder). Wie groß ist der Abstand eines Blattes von der Wurzel im besten und im schlimmsten Fall?

Tutoraufgabe 2

a) Geben Sie ein Verfahren an, um zu entscheiden, ob ein vorgelegter Graph bipartit (also zweifärbbar) ist.

b) Zeigen Sie, daß jeder Graph mit maximalem Grad Δ immer $(\Delta + 1)$-partit ist (das schließt nicht aus, daß er nicht etwa auch mit weniger Farben gefärbt werden könnte, die Farbklassen dürfen leer sein).

Hinweis: *Die als freiwillig gekennzeichneten Aufgaben werden nicht bewertet und sind für die Klausurzulassung ohne Bedeutung.*

Freiwillige Aufgabe 1

Es sollen n Jobs $\{1, \ldots, n\}$ hintereinander abgearbeitet werden. Erschwerend kommt hinzu, daß manche Jobs vor anderen fertig sein müssen. Dieses formalisieren wir dadurch, daß wir eine gerichtete Kante $a = (i, j)$ einführen, wenn Job i vor Job j fertig sein soll. Wir betrachten also einen Digraphen, der die *Präzedenzrelation* der Jobs beschreibt.

a) Beschreiben Sie ein Verfahren um zu entscheiden, ob solch ein Digraph konsistent ist, daß er also keine (gerichteten) Zyklen enthält.

b) Angenommen, wir haben in a) festgestellt, daß es keine Zyklen gibt, der Digraph also ein DAG ist (directed acyclic graph). Nun wollen wir die Jobs so nummerieren (eine Permutation π der Jobs finden), daß, wann immer ein Job i vor dem Job j ausgeführt werden muß, weil es einen gerichteten Pfad von i nach j gibt, dieser Job i auch vor Job j ausgeführt wird, also $\pi(i) < \pi(j)$. Geben Sie ein Verfahren an, eine solche Permutation π zu finden.

   Bem: Eine Permutation wie in b) bezeichnet man als *topologische Nummerierung.*

Freiwillige Aufgabe 2

Sei ein Graph $G = (V, E)$ gegeben, $|V| = n + 1, |E| = m$. Jeder Spannbaum $T(G)$ enthält genau n der insgesamt m Kanten. Man erhält einen solchen Spannbaum etwa, indem man in einer beliebigen Reihenfolge alle Kanten des Graphen ausprobiert und jeweils solche Kanten behält, die zusammen mit den bereits zuvor behaltenen Kanten *keinen* Kreis schließen.
Offenbar ist das Ergebnis abhängig von der Wahl der Reihenfolge. *Wieviele* verschiedene Spannbäume gibt es?

Hinweise: Für eine $m \times n$-Matrix A gilt die Binet-Cauchy Formel:

$$\det A^T A = \sum_{S \subset \{1,\dots,m\}, |S|=n} \det A_S^T A_S.$$

Dabei sei A_S die $n \times n$-Matrix, die aus A entsteht, indem man alle Zeilen streicht, deren Nummern nicht in S liegen. (Wir identifizieren ggf. eine Kante mit ihrer Nummer in $\{1, \dots, m\}$.)
Wir erzeugen nun die $m \times (n + 1)$-Matrix $\hat{A}$ aus der $m \times (n + 1)$-Inzidenzmatrix von G, indem dort in jeder Zeile jeweils einer der beiden Einträge mit dem Wert $+1$ durch -1 ersetzt wird. Die obige $m \times n$-Matrix A entsteht jetzt, indem man aus $\hat{A}$ die erste Spalte wegstreicht.
Nun erkennt man, daß in der obigen Formel der Summationsindex S alle n-elementigen Teilmengen der Kanten von G durchläuft, insbesondere also auch jene, die Spannbäume von G sind.
Erfreulicherweise sind (siehe Teilaufgabe a) von allen Summanden genau die Summanden gleich 1, die zu einem Summationsindex gehören, der einem Spannbaum korrespondiert, alle anderen Summanden sind Null. Also zählt die obige Formel genau die Anzahl der Spannbäume von G.

a) Beweisen Sie, daß tatsächlich $\det A_S^T A_S$ genau dann gleich 1 ist, wenn S einem Spannbaum korrespondiert, und anderenfalls gleich Null ist.

b) Geben Sie mit Hilfe der neu gewonnenen Formel einen alternativen Beweis für den Satz von Cayley, indem Sie die Spannbäume des vollständigen Graphen K_n zählen.

c) Für ganz ehrgeizige Studenten: Habe nun jede Kante e ein (positiv ganzzahliges) Gewicht $w(e)$. Nun kann man einem Spannbaum, der durch eine Kantenmenge S gegeben ist, das Gewicht $w(S) := \sum_{e \in S} w(e)$ zuordnen und fragen, wie viele Spannbäume es vom Gewicht $k \in \mathbb{N}_0$ gibt. *Tip:* Multiplizieren Sie jede Spalte von A^T auf der linken Seite der obigen Formel mit $x^{w(e)}$ der zu dieser Spalte gehörigen Kante e. Entsprechend werden die A_S^T auf der rechten Seite modifiziert. Man erhält so eine Identität für eine Funktion von x, genauer ein Polynom $\sum_{k\in\mathbb{N}_0} p_k x^k$. Zeigen Sie durch Koeffizientenvergleich, daß die Koeffizienten p_k des Polynoms die gesuchten Größen sind.

d) Wir haben in der Datei `weighted_graph` auf der Übungsseite einen gewichteten Graphen abgelegt (Anleitung zur Benutzung mit Maple siehe Computeraufgaben). Berechnen Sie die linke Seite der Identität mit Hilfe von Maple und verifizieren Sie, daß der gewichtete Graph genau einen Spannbaum mit Gewicht 28, aber drei Spannbäume mit Gewicht 16 besitzt. Wieviele Spannbäume mit Gewicht 22 besitzt dieser Graph?

***Hinweise** zur Bearbeitung der Computeraufgaben finden Sie auf der Web-Seite zu den Übungen.*

Computeraufgabe 1 (10 Punkte)

Auf der Übungsseite
`wwwmayr.informatik.tu-muenchen.de/lehre/2000WS/ds/uebung/`
finden Sie drei Graphen in den Dateien
`graph1,graph1_explicit`
`graph2,graph2_explicit`
`graph3,graph3_explicit`.
Alle diese Dateien können in Maple eingelesen werden mittels
`>read graph1:`
etc.
Im ersten Dateityp steht der jeweilige Graph G_i als Maple-Datenstruktur aus dem Paket `networks` (vgl. `>?networks`) und kann mit den Routinen aus diesem Paket leicht manipuliert werden.
Am meisten lernen Sie aber wohl, wenn Sie selbst eine Breitensuche (BFS) programmieren. Deshalb stehen im zweiten Dateityp jeweils nur die Knotenmenge V_i (immer [50]) und die Menge E_i der Kanten. Dieser zweite Typ von Datei kann auch leicht von einem selbstgeschriebenen Java- oder C++-Programm eingelesen werden.
Snippet weist Ihnen zufällig eine Nummer i zu. Der Graph G_i wird im folgenden als *Ihr Graph* bezeichnet.

a) Bestimmen Sie die Größe der größten Zusammenhangskomponente Ihres Graphen.

b) Bestimmen Sie die Abstände der Knoten 15 und 25 in Ihrem Graphen (falls die beiden Knoten in verschiedenen Zusammenhangskomponenten liegen sollten, tragen Sie bitte '-1' ein).

c) Bestimmen Sie die Anzahl von Pfaden der Länge 3 (bzw. 4, je nach Aufgabenstellung) zwischen den Knoten 3 und 29. Dabei ist ein Pfad der Länge k eine Folge von $k+1$ paarweise durch Kanten verbundenen Knoten, so daß zwei aufeinander folgende Knoten stets verschieden sind ('Stehenbleiben verboten'). *Tip:* Benutzen Sie die Adjazenzmatrix Ihres Graphen!

Computeraufgabe 2 (10 Punkte)

a) Wieviele verschiedene Wälder mit jeweils genau zwei (*sic*) nichtleeren Bäumen gibt es auf n Knoten? (Die Knoten sollen also in zwei disjunkte, nichtleere Bäume partitioniert werden. Isomorphe Kopien werden *nicht* identifiziert.)

b) Wieviele bipartite Graphen mit m Kanten gibt es, wenn die beiden Teilmengen V_1, V_2 der Bipartition gleich groß sind, also $|V_1| = |V_2| = n$.

c) Wieviele einfache Kreise der Länge r gibt es im vollständigen Graphen K_n (auf n Knoten)? *Tip:* Achten Sie darauf, nichts doppelt zu zählen.

   Ein einfacher Kreis der Länge r ist ein Subgraph mit genau r Kanten $e_1, \ldots e_r$, so daß die Kante mit Nummer k jeweils genau einen Knoten mit den Kanten mit Nummern $k-1$ und $k+1$ (modulo r) gemeinsam hat. Alle r an einem solchen Kreis beteiligten Knoten sind verschieden.