Johannes Blömer
u.v.a.m.

Paderborn, den 11. Juli 2004

Musterlösungen zur Vorlesung

Datenstrukturen und Algorithmen

SS 2004

Blatt 11

AUFGABE 1:
Sei A_G die Adjanzenzmatrix des gerichteten Graphen $G = (V, E)$. Mit A_G^T bezeichnen wir die Adjazenzmatrix von G^T. Aus der Definition von G^T folgt, dass der Eintrag in der i-ten Zeile und j-ten Spalte von A_G^T genau dann 1 ist, wenn der Eintrag in der j-ten Zeile und i-ten Spalte von A ebenfalls 1 ist. Die Matrix A_G^T ist also die *Transponierte* der Matrix A_G. Daher auch unsere Notation A_G^T, die aus der linearen Algebra bekannt sein sollte. Um A_G^T zu berechnen, müssen wir alle Indexpaare $(i, j), 1 \leq i, j \leq |V|$ einmal durchlaufen, den entsprechenden Eintrag a_{ji} in A_G bestimmen und den Eintrag in die i-te Zeile und j-te Spalte von A_G^T übertragen. Pro Matrixeintrag haben wir damit konstanten Aufwand und benötigen insgesamt Zeit $\mathcal{O}(|V|^2)$.

Mit Adj_T bezeichnen wir das Array, das die Adjazenzlisten des Graphen G^T enthält. Die Adjazenzliste des Knoten u im Graphen bezeichnen wir mit $Adj_T[u]$. Nun ist der Knoten v in der Adjazenzliste $Adj_T[u]$ genau dann enthalten, wenn u in der Adjazenzliste $Adj[v]$ des Graphen G enthalten ist. Um nun Adj_Z zu berechnen, durchlaufen wir einmal die Adjazenzlisten $Adj[v]$ für alle Knoten in V. Tritt dann u in $Adj[v]$ auf, so nehmen wir v in $Adj_T[u]$ auf. Diese Algorithmus berechnet das Array Adj_T korrekt. Um Adj_T zu berechnen, müssen wir jede Liste $Adj[v]$ betrachten und benötigen für jeden Eintrag in $Adj[v]$ konstanten Zeit. Insgesamt benötigen wir damit Zeit $\mathcal{O}(|V| + |E|)$ um Adj_T aus Adj zu berechnen.

AUFGABE 2:
Der Breitensuchbaum ist in Abbildung 1 gegeben. Für die Werte $\pi[u]$ und $d[u]$ erhalten wir folgende Tabelle, in der jeder Knoten mit seinem Schlüssel identifiziert wurde.

Knoten	$\pi[u]$	$d[u]$
3	NIL	0
2	3	1
4	3	1
1	2	2
5	2	2

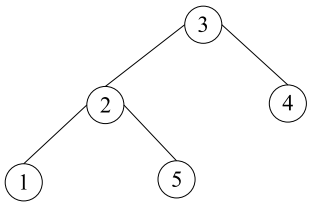

Abbildung 1: Breitensuchbaum zu Aufgabe 2

AUFGABE 3:
Unser Algorithmus COLORABLE zur Entscheidung, ob ein Graph $G = (V, E)$ 2-färbbar ist, arbeitet in drei Schritten.

COLORABLE(G)

1. Schritt Führe eine Breitensuche auf dem Graphen $G = (V, E)$ durch.

2. Schritt Färbe alle Knoten u, für die $d[u]$ gerade ist, blau. Die Knoten u für die $d[u]$ ungerade ist, färbe rot.

3. Schritt Durchlaufe alle Kanten (u, v) aus E und überprüfe, ob die Knoten u und v unterschiedlich gefärbt sind. In dieses der Fall Ausgabe "2-färbbar", sonst Ausgabe "nicht 2-färbbar".

Überlegen wir uns zunächst, dass die Laufzeit dieses Algorithmus $\mathcal{O}(|V| + |E|)$ ist. Wie in der Vorlesung gesehen, benötigen wir für die Breitensuche im 1. Schritt Zeit $\mathcal{O}(|V| + |E|)$. Im zweiten Schritt müssen wir alle Knoten einmal betrachten und die Farbe bestimmen. Dieses kann pro Knoten in konstanter Zeit durchgeführt werden. Insgesamt erfordert der 2. Schritt damit Zeit $\mathcal{O}(|V|)$. Für den 3. Schritt werden alle Kanten einmal durchlaufen. Dabei erfordert der Test, ob die Knoten der Kante unterschiedlich gefärbt sind, konstante Zeit. Für den 3. Schritt benötigen wir somit insgesamt Zeit $\mathcal{O}(|E|)$. Damit ist gezeigt, dass die Laufzeit von Algorithmus COLORABLE $\mathcal{O}(|V| + |E|)$ beträgt.
Nun zur Korrektheit des Algorithmus COLORABLE. Dazu zeigen wir, dass ein Graph G dann und nur dann 2-färbbar ist, wenn die von COLORABLE im 2. Schritt berechnete Färbung eine 2-Färbung von G ist. Zusammen mit dem 3. Schritt von COLORABLE folgt dann die Korrektheit.
Ist die von COLORABLE berechnet Färbung eine 2-Färbung, so ist G natürlich auch 2-färbbar.
Nehmen wir nun an, dass der Graph G 2-färbbar ist. Sei $F : V \to \{\text{blau}, \text{rot}\}$ eine Funktion, die eine korrekte 2-Färbung von G beschreibt. Betrachten wir den Breitensuchbaum T den COLORABLE im 1. Schritt berechnet. Sei u die Wurzel des Baums. Wir können annehmen, dass $F(u) = \text{blau}$ gilt. Ist nämlich $F(u) = \text{rot}$, so können wir sämtliche Knoten von G umfärben und erhalten dann eine 2-Färbung von G, in der u blau gefärbt ist. Alle Knoten $v \in V$ mit $d[v] = 1$ sind nun durch eine Kante in G mit u verbunden. Diese müssen also alle durch F rot gefärbt werden. Dann aber müssen alle Knoten mit $d[v] = 2$ durch F wieder blau gefärbt werden, denn sie sind alle durch eine Kante mit einem rot gefärbten Knoten w mit $d[w] = 1$ verbunden. Durch vollständige Induktion über $d[v]$ folgt nun, dass $F(v) = \text{blau}$ für alle Knoten v mit $d[v] = 0 \mod 2$ und $F(v) = \text{rot}$ für alle Knoten v mit $d[v] = 1 \mod 2$. Das heisst aber, dass die von COLORABLE im 2. Schritt berechnete Färbung genau die durch F gegebene Färbung ist. Dieses ist aber eine 2-Färbung. Damit berechnet COLORABLE im 2. Schritt für jeden 2-färbbaren Graphen auch eine 2-Färbung. Wie oben schon argumentiert, folgt nun, dass COLORABLE korrekt ist.

AUFGABE 4:
Der Tiefensuchbaum ist in Abbildung 2 gegeben.

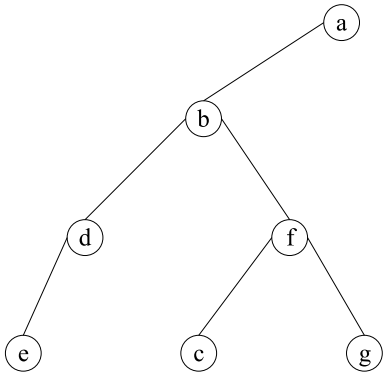

Abbildung 2: Tiefensuchbaum zu Aufgabe 4

AUFGABE 5:
Die topologische Sortierung ist in Abbildung 3 gegeben.

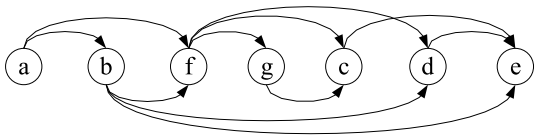

Abbildung 3: Topologische Sortierung zu Aufgabe 5