Teillösungen einiger Präsenzaufgaben von Übungsblatt 4

Lösungen zu Aufgabe 2
Die hier angegebenen Schranken sind nicht unbedingt scharf!

(a) $A[i,j] \neq 0$ (Zeit $\mathcal{O}(1)$). Bzw. $j \in L_i$ (Zeit $\mathcal{O}(\min\{m,n\})$).

(b) A symmetrisch, d.h., $\forall i,j \in V : A[i,j] = A[j,i]$ (Zeit $\mathcal{O}(n^2)$).
Bzw. $\forall i,j \in V : j \in L_i \Leftrightarrow i \in L_j$ (Zeit $\mathcal{O}(m \cdot n)$). Dabei zunächst Aufwand $\max\{m,n\}$ zum "Anfassen" aller Listeneinträge (insgesamt m) aller n Listen, sowie nach (a) jeweils Aufwand $\min\{m,n\}$, um für $i \in L_j$ auch in Liste L_i zu prüfen, ob $j \in L_i$. Zusammen also $\max\{m,n\} \cdot \min\{m,n\} = m \cdot n$.

(c) Hauptdiagonale enthält nur Nullen, d.h., $\forall i : A[i,i] = 0$ (Zeit $\mathcal{O}(n)$).
Bzw. $\forall i \in V : i \notin L_i$ (Zeit $\mathcal{O}(\max\{m,n\}) = \mathcal{O}(n+m)$).

(d) Jede Zeilensumme ist $\leq c$ (Zeit $\mathcal{O}(n^2)$).
Bzw. $\forall i \in V : |L_i| \leq c$ (Zeit $\mathcal{O}(n)$, sofern Länge einer Liste direkt abrufbar, sonst Zeit $\mathcal{O}(cn)$).

Lösungen zu Aufgabe 4

(a) BFS und DFS mit als Adjazenzlisten gegebenen Graphen haben Laufzeit $\mathcal{O}(n+m)$. Beide ermitteln insbesondere für jeden Knoten i einmal alle seine Nachbarn (und fügen diese gewissen Listen hinzu, oder verzweigen für jeden von ihnen rekursiv). Bei der Adjanzenzlistendarstellung sind dabei die Nachbarknoten sofort durch Zugriff auf die Liste L_i verfügbar, so dass insgesamt Aufwand $\mathcal{O}(\max\{m,n\} = \mathcal{O}(m+n))$ anfällt. Ist der Graph jedoch als Adjazenzmatrix A gegeben, so muss zum Ermitteln der Nachbarn von Knoten i die gesamte i-te Zeile in A überprüft werden, d.h., für jeden Knoten fällt Aufwand n an, insgesamt also $\mathcal{O}(n^2)$. Selbst ein Preprocessing, welches zunächst die Nicht-Null-Einträge in A ermittelt und später effizient bereitstellt, benötigt bereits Aufwand $\Omega(n^2)$, da jeder Eintrag in A überprüft werden muss.

(b) Die BFS geht Level für Level abwärts, und hält stets zunächst alle Knoten des Levels $\ell - 1$ in der Queue, bevor der erste Knoten auf Level ℓ besucht wird. Damit hält die Queue kurz vor Ende der BFS alle Knoten des vorletzen Levels (bei Höhe h also $2^{h-1} = \frac{1}{4}2^{h+1} \geq \frac{1}{4}n = \frac{1}{4}(2^{h+1}-1)$ viele). Daher Speicherbedarf $\Theta(n)$.

Die DFS hat üblicherweise Speicherbedarf $\Omega(n)$ allein schon deshalb, weil zu jedem Knoten ein Farbwert gespeichert werden muss. Die DFS kann aber in einem binären Baum der Höhe $h = \lceil \log_2 n \rceil$ auch anders implementiert werden. Zum Beispiel als Pre-Order-Treewalk. Der maximale Speicherbedarf ist daher die maximale Rekursionstiefe dieses Treewalks, also gleich der Höhe des Baums. Anders argumentiert: Die DFS kann durch einen fortlaufend aktualisierten Vektor v wechselnder Länge $\leq h$ beschrieben werden, dessen Einträge 'links' oder 'rechts' den Weg von der Wurzel zum aktuellen Knoten codieren, und der ähnlich wie beim binären Hochzählen aktualisiert wird. Derartige Varianten der DFS auf einem binären Baum haben damit Speicherplatzbedarf $\mathcal{O}(\log n)$.