Übungen zur Vorlesung

Algorithmen und Datenstrukturen

Übungsblatt 11

Arbeitsgruppe Kryptographie und Komplexitätstheorie
Prof. Dr. Marc Fischlin
Dr. Christian Janson
Patrick Harasser
Felix Rohrbach

Sommersemester 2019
Veröffentlicht am: 29.06.2019
Abgabe am: 12.07.2019, 12:00 Uhr

P1 (Gruppendiskussion)

Nehmen Sie sich etwas Zeit, um die folgenden Fachbegriffe in einer Kleingruppe zu besprechen, sodass sie anschließend in der Lage sind, die Begriffe dem Rest der Übungsgruppe zu erklären:

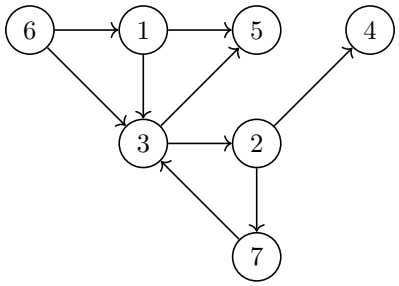
(a) Adjazenzmatrix und Adjazenzliste
(b) Gerichtete und ungerichtete Graphen
(c) Breadth-First Search (BFS) und Depth-First Search (DFS)
(d) Starke Zusammenhangskomponenten

P2 (Darstellung von Graphen)

(a) Geben Sie die Adjazenzmatrix und Adjazenzliste eines vollständigen binären Baumes mit 15 Knoten an. Nummerieren Sie die Knoten von oben nach unten und von links nach rechts.
(b) Geben Sie für Adjazenzmatrizen und Adjazenzlisten die asymptotische Laufzeit an, um zu überprüfen, ob eine Kante zwischen zwei Knoten liegt.
(c) Geben Sie je einen Algorithmus an, um eine Adjazenzmatrix in eine Adjazenzliste und umgekehrt umzurechnen.

P3 (Breiten- und Tiefensuche)

Gegeben sei folgender gerichteter Graph G:



(a) Führen Sie die Breitensuche auf G ausgehend vom Knoten 6 aus. Wenn mehrere Knoten zur Wahl stehen, wählen Sie zuerst immer den Knoten mit dem kleinsten Schlüssel aus.
Füllen Sie dazu die gegebene Tabelle aus. Die Tabelle enthält eine separate Zeile für jede Iteration der **while**-Schleife im Algorithmus. Geben Sie für jede Iteration der Schleife an, welcher Knoten u gerade bearbeitet wird, welche Nachbarn v in dem Schritt entdeckt werden, und welche Elemente sich in der Queue Q am Ende der Iteration befinden. Geben Sie anschließend für jeden Knoten die Distanz zum Ursprungsknoten 6, sowie den Vorgängerknoten, welche aus dem Verfahren hervorgehen, an.

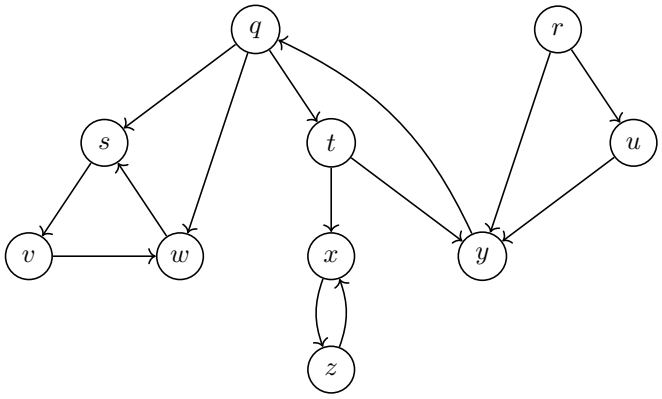
Iteration	u	v	Q
0	-	□	[6]

(b) Führen Sie die Tiefensuche auf G aus. Wenn mehrere Knoten zur Wahl stehen, wählen Sie zuerst immer den Knoten mit dem kleinsten Schlüssel aus.
Füllen Sie dazu die gegebene Tabelle aus. Die Tabelle enthält eine Zeile für jeden Knoten im Graphen. Geben Sie für jeden Knoten die jeweilige Entdeckungszeit, Abschlusszeit, und den Vorgängerknoten, welche aus dem Verfahren hervorgehen, an.

Knoten	Entdeckungszeit	Abschlusszeit	Vorgängerknoten
1	1	12	`nil`

P4 (Starke Zusammenhangskomponenten)

(a) Wie kann sich die Anzahl der starken Zusammenhangskomponenten ändern, wenn eine neue Kante hinzugefügt wird?
(b) Führen Sie den Algorithmus zur Bestimmung von starken Zusammenhangskomponenten auf diesem Graphen aus. Wählen Sie dabei in der Tiefensuche immer das lexigraphisch kleinste Element, falls Sie zwischen mehreren Knoten wählen müssen.



P5 (Eulerkreis)

Ein Eulerkreis ist ein Zyklus auf einem (stark) verbundenen, gerichteten Graph, der jede Kante genau einmal besucht.

(a) Begründen Sie: Damit ein Eulerkreis existieren kann, muss für jeden Knoten die Anzahl der ankommenden Kanten gleich der Anzahl der abgehenden Kanten sein.

(b) Ihnen wird ein Algorithmus gegeben, der beginnend einem zufälligen Knoten immer eine zufällige, noch nicht besuchte abgehende Kante wählt, diese der Liste der besuchten Kanten hinzufügt, und dies vom Zielknoten wiederholt, bis keine unbesuchte ausgehende Kante am aktuellen Knoten mehr existiert. Warum erzeugt dieser Algorithmus auf jedem Graphen mit einem Eulerkreis einen Kreis, der jede Kante maximal einmal besucht, auf diesem Graphen?

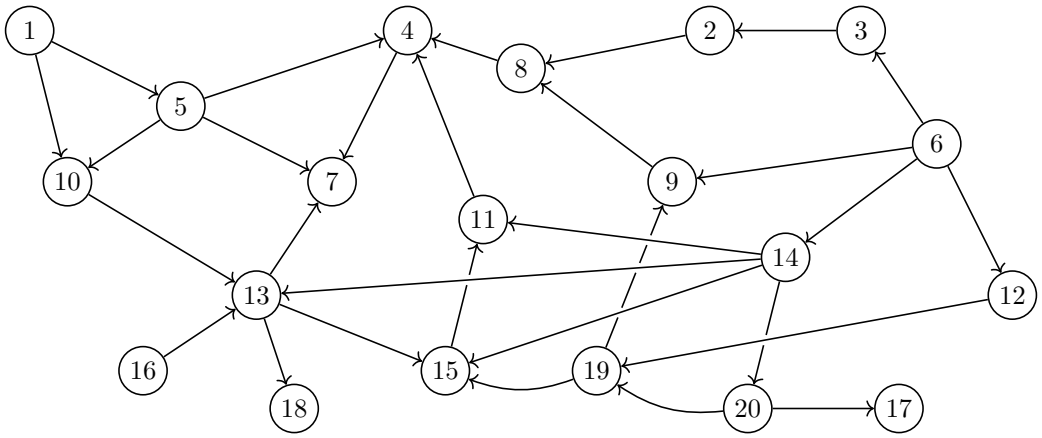
(c) Wie muss man den Algorithmus aus (b) ergänzen, damit er einen Eulerkreis erzeugt?

Hausübungen

In diesem Bereich finden Sie die theoretischen Hausübungen von Blatt 11. Bitte beachten Sie die allgemeinen Hinweise zu den Hausübungen und deren Abgabe im Moodle-Kurs! **Denken Sie bitte daran, dass Ihre Lösungen nachvollziehbar und entsprechend ausführlich dargestellt und begründet werden sollen.**
Bitte reichen Sie Ihre Abgabe bis spätestens **Freitag, 12.07.2019, um 12:00 Uhr** ein. Verspätete Abgaben können **nicht** berücksichtigt werden.

H1 (Breitensuche und Tiefensuche) (2.5+2.5 Punkte)

Gegeben sei folgender gerichteter, azyklischer Graph G:



(a) i) Führen Sie die Breitensuche auf G ausgehend vom Knoten 6 aus. Wenn mehrere Knoten zur Wahl stehen, wählen Sie zuerst immer den Knoten mit dem kleinsten Schlüssel aus.
Füllen Sie dazu die erste Tabelle im Dokument `Vorlage_Breiten-Tiefensuche.pdf` aus. Die Tabelle enthält eine separate Zeile für jede Iteration der **while**-Schleife im Algorithmus. Geben Sie für jede Iteration der Schleife an, welcher Knoten u gerade bearbeitet wird, welche Nachbarn v in dem Schritt entdeckt werden, und welche Elemente sich in der Queue Q am Ende der Iteration befinden. Geben Sie anschließend für jeden Knoten die Distanz zum Ursprungsknoten 6, sowie den Vorgängerknoten, welche aus dem Verfahren hervorgehen, an.
Bitte beachten Sie: Die Tabelle muss vollständig ausgefüllt werden. Insbesondere werden leere Zellen als Fehler gewertet. Verwenden Sie das Symbol " □ " um anzuzeigen, dass keine neuen Knoten im entsprechenden Schritt entdeckt wurden.

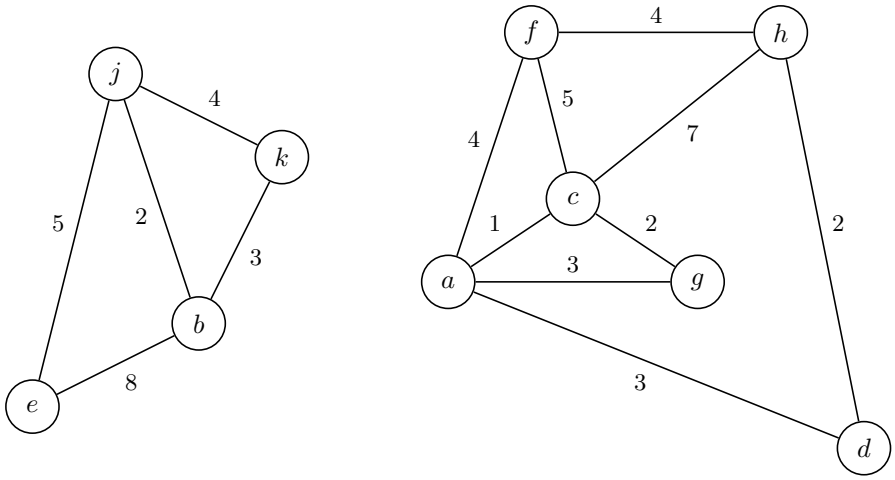
ii) Zeichnen Sie den BFS-Baum G^6_{pred}.

(b) i) Führen Sie die Tiefensuche auf G aus. Wenn mehrere Knoten zur Wahl stehen, wählen Sie zuerst immer den Knoten mit dem kleinsten Schlüssel aus.
Füllen Sie dazu die zweite Tabelle im Dokument `Vorlage_Breiten-Tiefensuche.pdf` aus. Die Tabelle enthält eine Zeile für jeden Knoten im Graphen. Geben Sie für jeden Knoten die jeweilige Entdeckungszeit, Abschlusszeit, und den Vorgängerknoten, welche aus dem Verfahren hervorgehen, an.

ii) Geben Sie die topologische Sortierung von G an, und zeichnen Sie den DFS-Wald G_{pred}.

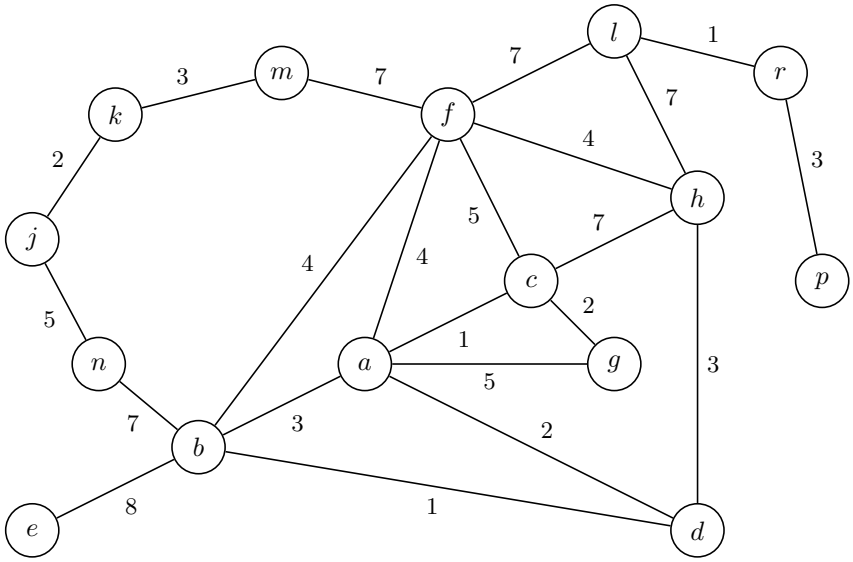
H2 (Kruskal und Prim) (2+3 Punkte)

(a) Führen Sie für den folgenden Graphen den Algorithmus von Kruskal aus. Kanten mit gleichem Gewicht sollen dabei alphabetisch sortiert bearbeitet werden (beispielsweise wird $\{b, j\}$ vor $\{c, g\}$ betrachtet).



Füllen Sie dazu die Tabelle im Dokument `Vorlage_Kruskal.pdf` aus. Die Tabelle enthält eine separate Zeile für jede Iteration der Schleife im Algorithmus. Sie sollen in jeder Zeile angeben, welche Kante $\{u, v\}$ im gegebenen Schritt betrachtet wird, das Gewicht $w(\{u, v\})$ von $\{u, v\}$, ob die Kante dem minimalen Spannbaum hinzugefügt wird oder nicht (j/n), sowie die Menge set(S) für jeden Knoten S des Graphen am Ende des betrachteten Schrittes. Zeichnen Sie anschließend den gefundenen minimalen Spannbaum, indem Sie dessen Knoten und Kanten im Graphen hervorheben.
Bitte beachten Sie: Die Tabelle muss vollständig ausgefüllt werden. Insbesondere werden leere Zellen als Fehler gewertet. Sie dürfen das Symbol " = " verwenden, wenn Sie den Inhalt der Zelle oberhalb der aktuellen Zelle in die aktuelle Zelle kopieren möchten (natürlich dürfen Sie den Inhalt auch nochmal abschreiben).

(b) Führen Sie für den folgenden Graphen den Algorithmus von Prim aus. Beginnen Sie bei Knoten A. Knoten mit gleichem Key-Wert sollen dabei alphabetisch sortiert bearbeitet werden.



Füllen Sie dazu die Tabellen im Dokument `Vorlage_Prim.pdf` aus. Beide Tabellen enthalten jeweils eine separate Zeile für jede Iteration der **while**-Schleife im Algorithmus. Geben Sie in jeder Zeile an, welcher Knoten u im gegebenen Schritt aus der Menge Q extrahiert wird, und die Key- und Pred-Werte s.`k` und s.`p` für jeden Knoten s des Graphen, sowie die Menge Q, nach dem betrachteten Schritt. Zeichnen Sie anschließend den gefundenen minimalen Spannbaum, indem Sie dessen Knoten und Kanten im Graphen hervorheben.
Bitte beachten Sie: Die Tabelle muss vollständig ausgefüllt werden. Insbesondere werden leere Zellen als Fehler gewertet. Sie dürfen das Symbol " = " verwenden, wenn Sie den Inhalt der Zelle oberhalb der aktuellen Zelle in die aktuelle Zelle kopieren möchten (natürlich dürfen Sie den Inhalt auch nochmal abschreiben).

H3 (Dreiecke in einem Graphen) (1.5+2+1.5 Punkte)

Ein *Dreieck* in einem ungerichteten Graphen $G = (V, E)$ ist eine Teilmenge $\{u, v, w\} \subseteq V$ paarweise verschiedener Knoten mit $\{u, v\}, \{u, w\}, \{v, w\} \in E$. Ziel dieser Aufgabe ist es einen Algorithmus zu erarbeiten, der einen endlichen, ungerichteten Graphen G mit n Knoten als Eingabe bekommt, und die Anzahl der Dreiecke in G ausgibt.

(a) Sei A die Adjazenzmatrix von G, also $A \in \mathbb{Z}^{n \times n}$ und

$$a_{ij} = \begin{cases} 1 & \text{falls } \{v_i, v_j\} \in E \\ 0 & \text{sonst.} \end{cases}$$

Für ein $r \geq 1$ bezeichne $a_{ij}^{(r)}$ den Eintrag an der Stelle (i, j) der Matrix A^r. Zeigen Sie, dass es in G genau $a_{ij}^{(r)}$ Wege der Länge r von v_i zu v_j gibt.

(b) Geben Sie einen Algorithmus GraphTriangles mit Laufzeit $O(n^3)$ an, der die Adjazenzmatrix eines endlichen, ungerichteten Graphen G als Eingabe bekommt, und die Anzahl der Dreiecke in G ausgibt. Falls Sie die Matrixmultiplikation als Operation benötigen, geben Sie expliziten Pseudocode dafür an.

(c) Beweisen Sie die Korrektheit und analysieren Sie die Laufzeit von GraphTriangles.