Algorithmische Mathematik:
Graphen & Anwendungen

Frühlingssemester 2018
P. Zaspel und I. Kalmykov

Universität Basel

Übungsblatt 4.

zu bearbeiten bis **Dienstag, 27.3.2018, 14:00 Uhr**.

Aufgabe 1. (Zusammenhang)

Sei $G = (V, E)$ ein ungerichteter Graph. Für alle $\emptyset \neq X \subsetneq V$ definieren wir die Hilfsgröße

$$\delta(X) := \{(x, w) \in E : x \in X, w \in V \setminus X\}.$$

Zeigen Sie folgende Äquivalenz:

$$\forall\, \emptyset \neq X \subsetneq V \colon \delta(X) \neq \emptyset \iff G \text{ zusammenhängend.}$$

(4 Punkte)

Aufgabe 2. (Topologische Ordnung)

Sei $G = (V, E)$ ein gerichteter Graph. Wir speichern G mit Adjazenzlisten, wobei zu jedem Knoten $v \in V$ zwei Listen geführt werden

- Adjazenzliste der Knoten $w \in \text{post}(v)$,
- Adjazenzliste der Knoten $r \in \text{pre}(v)$.

Zeigen Sie, dass mithilfe dieser Datenstruktur in linearer Zeit $\mathcal{O}(|V| + |E|)$ entweder eine topologische Ordnung oder ein Kreis in G gefunden werden kann.
Bemerkung. Die Definition der topologischen Ordnung für einen gerichteten Graphen finden Sie in der Aufgabe 3 auf dem 3. Blatt.

(4 Punkte)

Aufgabe 3. (Tiefensuche/Breitensuche)

Gegeben sei der Graph $G = (V, E)$ aus Abbildung 1.

a) Geben Sie die Besuchsreihenfolge der Knoten an, wenn im Graphen G aus Abbildung 1 eine *Tiefensuche* beginnend im Knoten v_2 mit Algorithmus 2.1 aus der Vorlesung durchgeführt wird. Skizzieren Sie zudem den DFS-Baum.

b) Geben Sie die Besuchsreihenfolge der Knoten an, wenn im Graphen aus Abbildung 1 eine *Breitensuche* beginnend im Knoten v_1 mit Algorithmus 2.1 aus der Vorlesung durchgeführt wird. Skizzieren Sie zudem den BFS-Baum.

Aufgabe 4. (Tiefensuche/Breitensuche)

Implementieren Sie Algorithmen zur Tiefensuche bzw. Breitensuche ausgehend von Algorithmus 1. Testen Sie Ihre Implementierung an dem Graphen aus Abbildung 1 mit Startknoten v_1. Plotten Sie die resultierenden DFS- und BFS-Bäume.
Bemerkung. Bei der Tiefensuche wird in der Zeile 3 derjenige Knoten $v \in Q$ ausgewählt, der zuletzt zu Q hinzugefügt wurde. In diesem Fall bezeichnet man die Implementierung für Q als einen LIFO-Stack (last in first out). Eine mögliche Umsetzung in Matlab ist

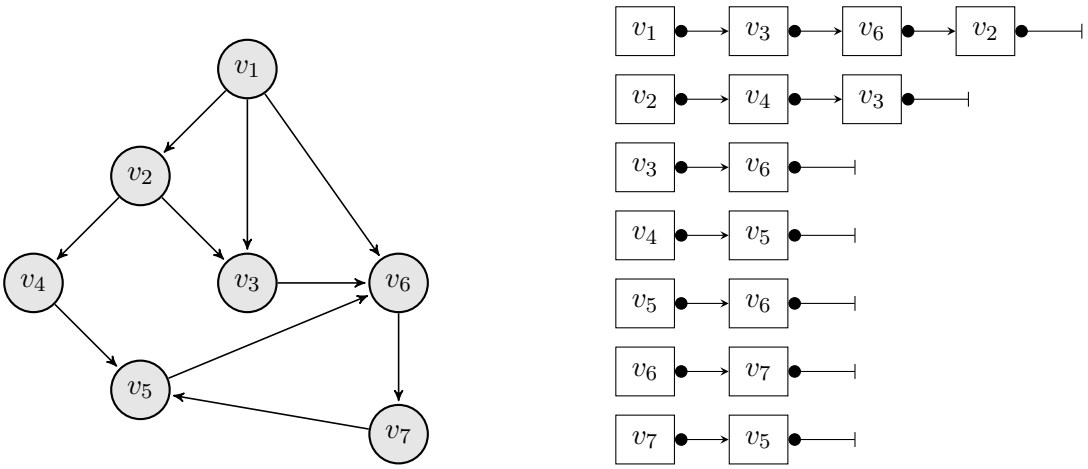

Abbildung 1: Graph G und Adjazenzliste.

1: % Hinzufügen von v
2: $Q = [Q, v]$
3: ... % Q wird nicht verändert
4: % Wähle w als das letzte Element in Q und entferne es aus Q
5: $w = Q\,[end]$
6: $Q = Q\,[1 : end - 1]$.

Bei der Breitensuche wird in der Zeile 3 derjenige Knoten $v \in Q$ ausgewählt, der zuerst zu Q hinzugefügt wurde. In diesem Fall bezeichnet man die Implementierung für Q als eine FIFO-Warteschlange (first in first out). Eine mögliche Implementierung in Matlab sieht wie folt aus

1: % Hinzufügen von v
2: $Q = [Q, v]$
3: ... % Q wird nicht verändert
4: % Wähle w als das erste Element in Q und entferne es aus Q
5: $w = Q\,[1]$
6: $Q = Q\,[2 : end]$.

Algorithmus 1 (Graphendurchmusterung)

Input: Graph $G = (V, E)$ und Startknoten $s \in V$
Output: gerichteter Baum $G' = (R, T)$ mit Wurzel s und $R = \{s\} \cup \text{post}^\star(s)$, $T \subseteq E$

1: $R \leftarrow \{s\}\,, Q \leftarrow \{s\}\,, T \leftarrow \emptyset$
2: **while** $Q \neq \emptyset$ **do**
3: Wähle $v \in Q$
4: **if** $\text{post}(v) \cap (V \setminus R) \neq \emptyset$ **then**
5: Wähle $w \in \text{post}(v) \cap (V \setminus R)$
6: $R \leftarrow R \cup \{w\}\,, Q \leftarrow Q \cup \{w\}\,, T \leftarrow T \cup \{e\}$
7: **else**
8: $Q \leftarrow Q \setminus \{v\}$
9: **end if**
10: **end while**
11: **return** (R, T)