Übungsblatt 4.

Bearbeiten bis: **Montag, 31.03.2025**

Aufgabe 1 (Gerade Anzahl Kanten). Jeder zusammenhängende Graph mit einer geraden Anzahl von Kanten lässt sich in paarweise kantendisjunkte Pfade der Länge 2 zerlegen.

Aufgabe 2 (Tiefensuche/Breitensuche | 4 Punkte). Gegeben sei der Digraph $G = (V, E)$ aus Abbildung 1.

(a) Geben Sie die Besuchsreihenfolge der Knoten an, wenn im Digraphen G aus Abbildung 1 eine *Tiefensuche* beginnend im Knoten v_2 durchgeführt wird, wobei nur die von v_2 erreichbaren Knoten besucht werden. Skizzieren Sie zudem den DFS-Digraphen.

(b) Geben Sie die Besuchsreihenfolge der Knoten an, wenn im Digraphen aus Abbildung 1 eine *Breitensuche* beginnend im Knoten v_1 durchgeführt wird, wobei nur die von v_1 erreichbaren Knoten besucht werden. Skizzieren Sie zudem den BFS-Digraphen.

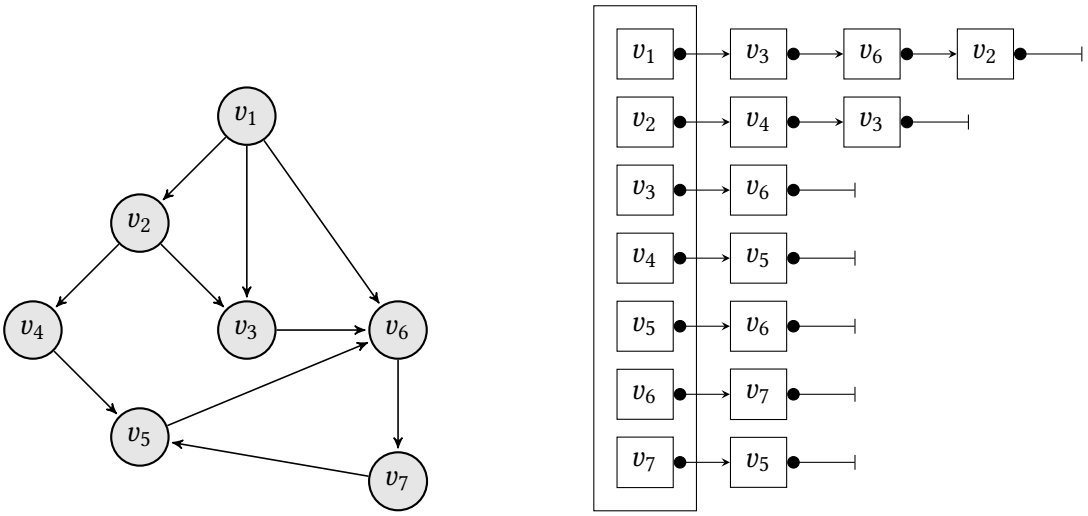


Abbildung 1: Digraph G und die zugehörige Adjazenzliste.

Aufgabe 3 ((Di-)Graph). Sei G ein Graph. Man, zeige, dass die Kanten von G so orientiert werden können, dass der entsprechende Digraph azyklisch ist.

Aufgabe 4 (Tiefensuche/Breitensuche | 4 Punkte). Mit der Implementierung der zwei Algorithmen für das Einfügen und Entfernen von Knoten aus der vorherigen Programmieraufgabe, kann man nun die Graphendurchmusterung implementieren. Konkret sollen die Breiten- und Tiefensuche realisiert werden.

Die Algorithmen für die Breitensuche und Tiefensuche sind wie folgt gegeben; der einzige Unterschied ist die Verwaltung der Warteschlange q.

Algorithmus search $(\mathbf{A}, q)$

Input: Nachbarschaftsmatrix $\mathbf{A}$, Warteschlange mit Anfangsknoten $q = [u]$
Output: eine Nachbarschaftsmatrix $\mathbf{A}'$ für den BFS/DFS-Baum

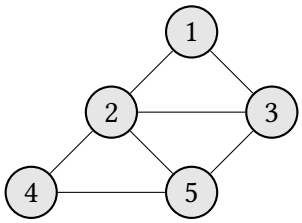
$r = [\,]$
while $q \neq [\,]$ **do**
  $n = \begin{cases} q[1] & \texttt{\% für Breitensuche} \\ q[end] & \texttt{\% für Tiefensuche} \end{cases}$
  if $\exists w = \mathbf{A}[i][n] \notin r \cup q$ mit i minimal **then**
    $q = [q, w]$
    $\mathbf{A}' = \text{addEdge}(\mathbf{A}', n, w)$
    $\mathbf{A} = \text{removeEdge}(\mathbf{A}, n, w)$
    $\mathbf{A} = \text{removeEdge}(\mathbf{A}, w, n)$ % falls der Graph ungerichtet ist
  else
    $r = [r, n]$
    $q = \begin{cases} q[2 : end] & \texttt{\% für Breitensuche} \\ q[1 : end - 1] & \texttt{\% für Tiefensuche} \end{cases}$
  end if
end while

Wenden Sie die Algorithmen auf die Graphen aus Abbildung 2 mit dem Anfangsknoten 1 an.

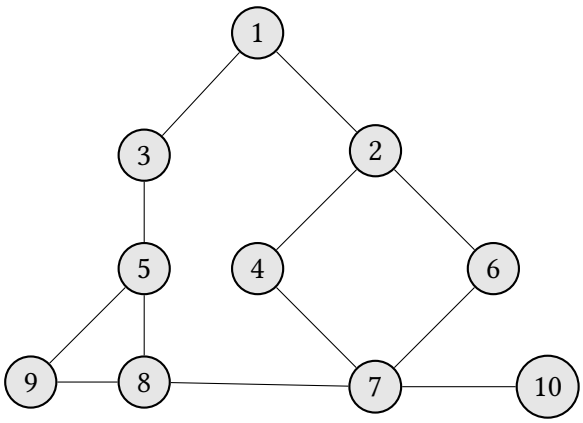
(a) Wie kann man den Algorithmus für die Breitensuche verbessern?

(b) Wie sehen die Nachbarschaftsmatrizen der Graphen G_1 und G_2 aus? Um Eindeutigkeit zu gewährleisten, sollen die Nachbarn in aufsteigender Reihenfolge in die Nachbarschaftsmatrizen eingefügt werden.

(c) Erstellen Sie die Nachbarschaftsmatrix der ensprechenden DFS- und BFS-Bäume.



(a) Graph G_1



(b) Graph G_2

Abbildung 2: Die Graphen G_1 und G_2.