Übungsblatt 3.

Bearbeiten bis: **Montag, 24.03.2025**

Aufgabe 1 (Zusammenhangskomponenten | 4 Punkte). Man zeige, dass ein Graph $G = (V, E)$, der $|E| > \binom{|V|-1}{2}$ erfüllt, zusammenhängend ist.

Aufgabe 2 (Knoten ungeraden Grades). Sei $G = (V, E)$ ein Graph sowie $u, v \in V$ die einzigen Knoten ungeraden Knotengrades in G mit $\deg(u) = 2l + 1$ und $\deg(v) = 2k + 1$, $l, k \in \mathbb{N}$. Zeigen Sie, dass ein Weg von u nach v in G existiert.

Aufgabe 3 (Eulersche Wege). Beweisen Sie folgende Variante des Satzes von Euler: Ein zusammenhängender Graph $G = (V, E)$ besitzt genau dann einen Weg in dem jede Kante aus E genau einmal vorkommt und dessen Anfangsknoten vom Endknoten verschieden sind, wenn es genau zwei Knoten mit ungeradem Knotengrad gibt.
Eine Anwendung dieses Satzes ist zum Beispiel das Haus von Nikolaus in Abbildung 1.

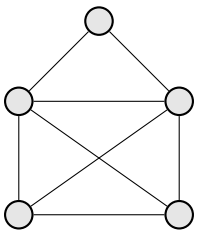

Abbildung 1: Das Haus von Nikolaus

Aufgabe 4 (Implementierung von Digraphen und Graphen | 4 Punkte). Eine Methode, um Digraphen und Graphen zu implementieren, ist die Realisierung der Adjazenzliste in Form einer Matrix. Sei hierzu k der maximale Ausgangsgrad der Knoten in $G = (V, E)$. Die Adjazenzliste kann dann in einer Matrix $\mathbf{A}$ der Grösse $(k + 1) \times |V|$ gespeichert werden. Für einen Knoten u mit Ausgangsgrad $d_u = |\text{post}(u)|$ werden die Nachfolger in den ersten d_u Zeilen der Spalte u gespeichert, also $\text{post}(u) = \{A[1][u], A[2][u], \dots, A[d_u][u]\}$. Alle anderen Elemente dieser Spalte werden auf 0 gesetzt, $A[d_u + 1][u] = \dots = A[k + 1][u] = 0$.

Um später mit Digraphen und Graphen arbeiten zu können, sollen auch die Funktionen für das Einfügen und das Entfernen von gerichteten Kanten implementiert werden. Diese sehen wie folgt aus:

Algorithmus addEdge $(\mathbf{A}, u, v)$ - Hinzufügen der Kante (u, v)

Input: Nachbarschaftsmatrix $\mathbf{A}$, Knoten u und Knoten v der neuen Kante
Output: neue Nachbarschaftsmatrix $\mathbf{A}$

if $A[i][u] \neq v$ für alle $i = 1, 2, \dots, d_u$ **then**
  $A[d_u + 1][u] = v$
end if

Algorithmus removeEdge $(\mathbf{A}, u, v)$ - Entfernen der Kante (u, v)

Input: Nachbarschaftsmatrix $\mathbf{A}$, Knoten u und Knoten v der zu entferndenden Kante
Output: neue Nachbarschaftsmatrix $\mathbf{A}$

if $A[i][u] = v$ für ein $i = 1, 2, \ldots, d_u$ **then**
  $A[i : d_u][u] = A[i + 1 : d_u + 1][u]$
end if

Um zu überprüfen, ob die Implementierung stimmt, sollen die implementierten Operationen auf den Graphen in Abbildung 2 angewendet werden:

(a) Wie sieht die Nachbarschaftsmatrix für den Graphen aus?

(b) Was passiert, wenn man die Kanten $\{2, 3\}$ und $\{4, 5\}$ entfernt?

(c) Wie sieht die Matrix aus, wenn man danach noch die Kante $\{1, 5\}$ hinzufügt?

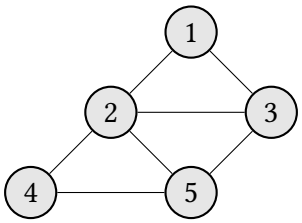

Abbildung 2: Graph G

Bemerkung. Der Graph ist ungerichtet. Wenn man also eine Kante $\{u, v\}$ hinzufügen oder entfernen möchte, muss man entsprechend die gerichteten Kanten (u, v) und (v, u) hinzufügen oder entfernen.

Hinweis. Um in `MATLAB` die Indizes der nicht-Null Einträge eines Vektors `v` zu bestimmen, kann man den Befehl `find(v ~= 0)` benutzen. Dieser Befehl kann auch verwendet werden, um die Indizes der Elemente, die eine gewisse Bedingung erfüllen, zu erhalten. Der Befehl `help find` gibt hierzu nähere Auskunft.