*Matthias Krause*
*Alexander Moch*

Mannheim, den 09.11.2020

**CS 307 Algorithmen und Datenstrukturen**, Herbstsemester 2020

# Lösungsskizze zum Übungsblatt 7

## AUFGABE 7.1:

Zeigen Sie, dass für einen ungerichteten Graphen $H = (W, F)$ die folgenden Charakterisierungen äquivalent sind:

**a)** $H$ ist ein Baum (zusammenhängend und kreisfrei),

**b)** $H$ ist zusammenhängend und $|F| = |W| - 1$,

**c)** $H$ ist kreisfrei und $|F| = |W| - 1$,

**d)** $H$ ist zusammenhängend und die Wegnahme jeder Kante macht $H$ unzusammenhängend,

**e)** $H$ ist kreisfrei und die Hinzunahme jeder neuen Kante erzeugt genau einen Kreis,

**f)** zwischen jedem Paar (verschiedener) Knoten von $H$ existiert genau ein Weg.

1. Wenn $H$ zusammenhängend ist, dann gilt: $|F| \geq |W| - 1$.
   *Beginnend mit einzelnen Knoten ($|W|$ Komponenten) betrachte die Kanten in beliebiger Reihenfolge und beobachte, dass jede Kante die Anzahl der Komponenten um höchstens 1 reduziert; also sind mindestens $|W| - 1$ Kanten nötig, damit am Ende nur eine Komponente existiert.*

2. Wenn $H$ kreisfrei ist, dann gilt: $|F| \leq |W| - 1$.
   *Traversiere den Graphen (z.B. mit Tiefsuche). Jede Kante führt entweder zu einem bisher unbesuchten Knoten oder erzeugt einen Kreis; nach Betrachtung von höchstens $|W|$ Kanten liegt also ein Kreis vor.*

3. $H$ ist kreisfrei genau dann, wenn zwischen jedem Paar (verschiedener) Knoten von $H$ höchstens ein Weg existiert.
   *Wenn es zwischen zwei Knoten $u$ und $v$ zwei nicht identische Wege $P$ und $P'$ gibt, dann gibt es eine Kante $(x, y) \neq (u, v)$, die auf $P$, aber nicht auf $P'$ liegt. Es existiert also mit $x \overset{P}{\rightsquigarrow} u \overset{P'}{\rightsquigarrow} v \overset{P}{\rightsquigarrow} y$ ein Weg, der $(x, y)$ nicht benutzt; also liegt $(x, y)$ auf einem Kreis. Umgekehrt existiert für jede Kreiskante $(x, y)$ ein Weg $x \rightsquigarrow y$, der $(x, y)$ nicht benutzt.*

*$a \to b$ folgt aus (1) und (2).*

*$a \to c$ folgt aus (1) und (2).*

*$b \to d$ folgt aus (1).*

*$c \to e$ folgt aus (2) und (3).*

*$d \to f$ folgt aus (3).*

*$e \to f$ folgt aus (3).*

*$f \to a$ folgt aus (3).*

## AUFGABE 7.2$^{\mathbf{K}}$:

Der Tiefensuche-Algorithmus DFS(-VISIT) weist jedem Knoten $v$ eine Entdeckungszeit $d[v]$ und eine Abschlusszeit $f[v]$ zu. Sei $G = (V, E)$ ein gerichteter Graph und $(u, v)$ mit $u \neq v$ eine Kante aus $E$. Geben Sie für jeden der folgenden Fälle eine (aufsteigend) sortierte Reihenfolge der Werte $d[u]$, $d[v]$, $f[u]$ und $f[v]$ an.

**a)** $(u, v)$ ist eine Tree-Kante,
$d[u] < d[v] < f[v] < f[u]$

**b)** $(u, v)$ ist eine Back-Kante,
$d[v] < d[u] < f[u] < f[v]$

**c)** $(u, v)$ ist eine Forward-Kante,
$d[u] < d[v] < f[v] < f[u]$

**d)** $(u, v)$ ist eine Cross-Kante.
$d[v] < f[v] < d[u] < f[u]$

## AUFGABE 7.3:

Betrachten Sie einen Knoten $v$ in einem gerichteten Graphen, der sowohl eingehende als auch ausgehende Kanten besitzt. Kann es passieren, dass bei der Ausführung des Algorithmus DFS dieser Knoten in einem (Tiefensuch-)Baum landet, der nur aus $v$ besteht? Begründen Sie Ihre Antwort.

*Ja, betrachte den Knoten* $2$ *in diesem Beispiel* (*Die Aufrufe von DFS-VISIT geschehen gemäß aufsteigender Nummern.*) $: 3 \to 2 \to 1$

## AUFGABE 7.4:

Um wie viel kann sich die Anzahl der starken Zusammenhangskomponenten eines gerichteten Graphen höchstens ändern, wenn eine einzige Kante hinzugefügt wird?

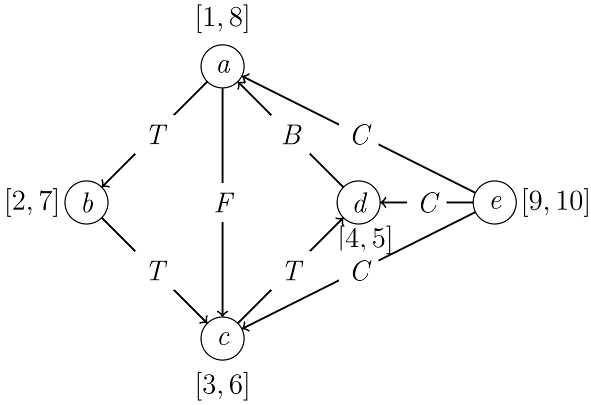
*Ein Graph mit $n$ Knoten hat wenigstens $1$ und höchstens $n$ starke Zusammenhangskomponenten. Das folgende Beispiel zeigt, dass eine Änderung von $n - 1$ tatsächlich möglich ist: Füge die Kante $n \to 1$ zum Graphen $1 \to 2 \to \cdots \to n$ hinzu.*

## AUFGABE 7.5$^{\mathbf{K}}$:

Der Graph $G = (V, E)$ über $V = \{a, b, c, d, e\}$ sei gegeben durch

$$E = \{(a, b), (a, c), (b, c), (c, d), (d, a), (e, a), (e, c), (e, d)\}.$$

Geben Sie die Bearbeitungsintervalle und die Klassifikation der Kanten in $T, B, F, C$-Kanten gemäß $DFS(G)$ an.
*Beachten Sie, dass die Knoten im Feld für $V$ und in den Listen adjazenter Knoten immer in alphabetischer Reihenfolge bearbeitet werden.*



[1, 8]
a
T
F
B
C
[2, 7]
b
d
C
e
[9, 10]
[4, 5]
T
T
C
c
[3, 6]