

Algorithms and Probability

Week 1

G09 - mkilic

19.II.2026

Overview

1. Organizational
2. Graph Theory Revisit
3. Zusammenhang
4. Exercises

Organizational

How to get bonus points?

- Minitests
- Theory Exercises
- Peer Grading
- Code Expert

About the exam

- A moodle part
- A written part
- Code Expert

Roadmap

1. Graphentheorie

- Zusammenhang
- Kreise
- Matchings
- Färbungen

2. W'keitstheorie

- Bedingte W'keit
- Unabhängigkeit
- (mehrere) Zufallsvariablen
- Diskrete Verteilungen
- Abschätzen von W'keiten
- Randomisierte Algorithmen

3. Algorithmen

- Lange-Bunte Pfade
- MaxFlow
- MinCut
- Kleinster umschliessender Kreis
- Konvexe Hülle

Graph Theory Revisit

Ein Sequenz von Knoten $\langle v_1, \dots, v_k \rangle$ heisst:

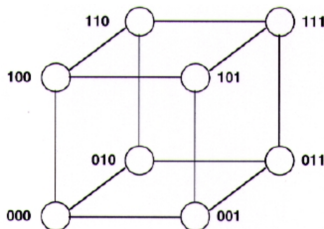
- **Weg:** falls für jedes $i \in \{1, \dots, k-1\}$ eine Kante von v_i nach v_{i+1} existiert
- **Pfad:** ein Weg wobei alle Knoten nur einmal vorkommen
- **Zyklus:** Weg mit $v_1 = v_k$
- **Kreis:** Zyklus der länge mindestens 3, wobei alle Knoten nur einmal vorkommen

Die Länge eines Weges ist die Anzahl der Schritte, hier $k-1$.

⁰Engl. *walk, path, Zyklus: closed walk, Kreis: cycle*

Graph Theory Revisit

- K_n : vollständiger Graph aus n Knoten
- C_n : Kreis aus n Knoten
- P_n : Pfad aus n Knoten
- Q_d : d -dimensionale Hyperwürfel



⁰Engl. *vollständig: complete* , *Hyperwürfel: hypercube*

Graph Theory Revisit

- **Bipartit:** Knoten kann man in zwei disjunkten Mengen teilen: $V = A \uplus B$ sodass Kanten nur zwischen A und B verlaufen.
- **Multigraphen:** Schleifen und Mehrfachkanten
- **Nachbarschaft:** $N(v) := \{u \in V \mid \{v, u\} \in E\}$
- **k-regulär:** $\forall v \in V : \deg(v) = k$

⁰Engl. *bipartite* , *multi-graphs* , *neighborhood* , *k-regular*

Pop Quiz

1. Für einen beliebigen Graphen $G = (V, E)$, $\sum_{v \in V} \deg(v) = ?$
2. Wann ist ein Kreis aus n Knoten bipartit?
3. Ist es möglich einen Graphen zu zeichnen, wobei 10 Knoten gerade und 7 Knoten ungerade Grad haben?

Pop Quiz: Solutions

1. $\sum_{v \in V} \deg(v) = 2|E|$, Handschlaglemma
2. Kreis aus n Knoten bipartit $\iff n$ ist gerade
3. Nein, für jeden Graphen gilt: Die Anzahl der Knoten mit ungeradem Grad ist gerade.
(*folgt aus Handschlaglemma*)

Zusammenhang

Zusammenhängend

$G = (V, E)$ ist zusammenhängend, wenn $\forall u, v \in V, u \neq v$ es gibt einen u - v -Pfad in G .

Wir fragen *Wie zusammenhängend ist mein Graph?*

Wieviele Knoten muss man mindestens löschen um den Zusammenhang zu zerstören?

⁰Engl. Zusammenhang: *connectivity*

Knoten-/Kanten-Zusammenhang

k-Zusammenhang

Ein Graph $G = (V, E)$ heisst *k-zusammenhängend*, wenn gilt

- $|V| \geq k + 1$
- $\forall X \subseteq V$ mit $|X| < k$ ist $G[V \setminus X]$ zusammenhängend

k-kanten-Zusammenhang

Ein Graph $G = (V, E)$ heisst *k-kanten-zusammenhängend*, wenn es gilt

- $\forall X \subseteq E$ mit $|X| < k$ ist $(V, E \setminus X)$ zusammenhängend

Intuitiv: Irgendeine Menge von *strikt weniger als k* Knoten / Kanten kann gelöscht werden. Der Graph bleibt zusammenhängend.

⁰Engl. Knotenzusammenhang: *vertex-connectivity*, Kantenzusammenhang: *edge-connectivity*

Separator

u-v-Separator:

Ist $G = (V, E)$, $u, v \in V$ und $X \subseteq V \setminus \{u, v\}$ eine Knotenmenge, für die u und v in verschiedenen ZHK's von $G[V \setminus X]$ liegen, dann ist X ein u - v -(Knoten^a)-Separator

^a(Kanten-Separator ähnlich definiert)

$G = (V, E)$ ist also genau dann k -zusammenhängend wenn jeder Separator mindestens Grösse k hat.

Ausnahme: Vollständige Graphen aus k Knoten sind $(k - 1)$ -zusammenhängend per Def.

⁰Engl. *Vertex/Edge-Separator, Vertex/Edge Cut, or separating set*

Satz von Menger

Satz von Menger

Sei $G = (V, E)$ ein Graph und $u, v \in V, u \neq v$ Dann

- a) G ist *k-zusammenhängend* $\iff$ Es gibt mindestens k intern-knotendisjunkte u - v -Pfade.
- b) G ist *k-kantenzusammenhängend* $\iff$ Es gibt mindestens k kantendisjunkte u - v -Pfade.

Man kann diese Relation auch mit Separatoren definieren:

Jeder u - v -(Kanten-)Separator hat Grösse mindestens $k \iff$ Es gibt mindestens k intern knotendisjunkte (kantendisjunkte) u - v -Pfade.

⁰intern-knotendisjunkt: alle Knoten verschieden ausser Anfang und Ende

Zusammenhang: Eigenschaften

- Knotenzusammengang $\leq$ Kantenzusammenhang $\leq$ minimaler Grad.
- Enthält G einen Knoten mit $\text{Grad} < k$ so ist G nicht k -zusammenhängend.
- Falls G k zusammenhängend, dann ist er auch $k - 1$ zusammenhängend.

Spezialfall: 1-zusammenhängend

Sei $G = (V, E)$ ein zusammenhängender Graph,

Artikulationsknoten

Ein Knoten $v \in V$ heisst *Artikulationsknoten* genau dann wenn $G[V \setminus \{v\}]$ nicht zusammenhängend ist.

Brücke

Eine Kante $e \in E$ heisst *Brücke* genau dann wenn der Graph $(V, E \setminus \{e\})$ nicht zusammenhängend ist.

⁰Engl. *Art.knoten*: cut vertex (articulation point), *Brücke*: cut edge (bridge)

Artikulationsknoten und Brücken

- Artikulationsknoten sind Zertifikat dafür, dass ein zusammenhängender Graph nicht 2-zusammenhängend ist.
- Brücken sind Zertifikat dafür, dass ein zusammenhängender Graph nicht 2-kanten-zusammenhängend ist.
- Ein Spannbaum muss immer alle Brücken des Graphen enthalten.
- Einen Artikulationsknoten kann es nur dann geben, wenn der Graph zusammenhängend ist.

Artikulationsknoten und Brücken

Lemma:

Sei $G = (V, E)$ ein zusammenhängender Graph. Ist $\{x, y\} \in E$ eine Brücke so gilt

$$\deg(x) = 1 \text{ oder } x \text{ ist Artikulationsknoten}$$

und analog für y .

Beweisskizze: Falls $\{x, y\}$ Brücke und $\deg(x) \neq 1 \Rightarrow x$ Artikulationsknoten

$\{x, y\}$ Brücke $\Rightarrow G' = (V, E \setminus \{x, y\})$ nicht zhg. insbesondere $\nexists$ x-y-Pfad in G' (1)

$\deg(x) \neq 1 \Rightarrow \exists u \neq y \in V$ s.d. $\{x, u\} \in (E \setminus \{x, y\})$ (2)

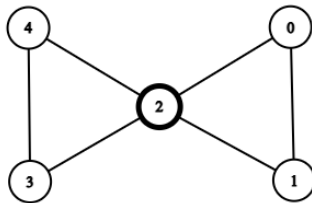
(1),(2) $\Rightarrow \nexists$ u-y Pfad in G'

$\Rightarrow$ Alle u-y-Pfade enthalten x

$\Rightarrow x$ ist ein Artikulationsknoten

Artikulationsknoten und Brücken

Die Umkehrung des Lemmas gilt im Allgemeinen nicht!



2 ist Artikulationsknoten aber hat weder Grad 1 noch eine inzidente Brücke.

Block-Zerlegung

Block

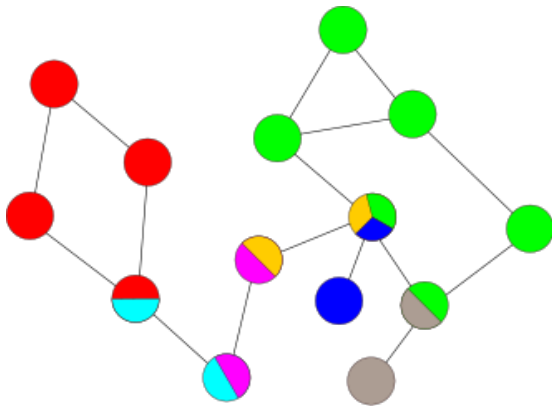
Sei $G = (V, E)$ ein zusammenhängender Graph. Für $e, f \in E$ definieren wir eine Relation durch:

$$e \sim f \begin{cases} e = f, \text{ oder} \\ \text{es gibt gemeinsamen Kreis durch } e \text{ und } f \end{cases}$$

Diese Relation ist eine Äquivalenzrelation. Wir nennen die Äquivalenzklassen *Blöcke*.

⁰Engl. Block: *biconnected component*, Block-Zerlegung: *block decomposition*

Block-Zerlegung



Fällt was auf an den mehrfarbigen Knoten?

Zwei Blöcke können sich nur in einem Artikulationsknoten überschneiden.

Äquivalenzrelation

Aus heiligem DiskMath Skript:

Definition 3.19. An *equivalence relation* is a relation on a set A that is reflexive, symmetric, and transitive.

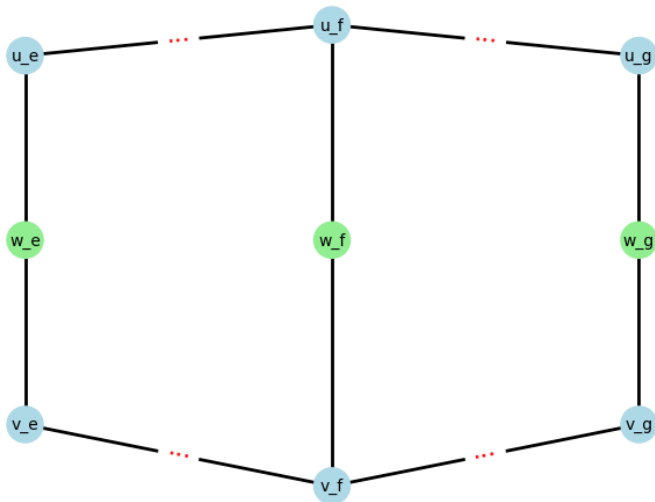
Die Relation $\sim$ ist offenbar reflexiv und symmetrisch per Definition
und Transitivität?

Beweis: Transitivität

Idee: Satz von Menger. Beweis:

1. Seien $e = \{u_e, v_e\}$, $f = \{u_f, v_f\}$, $g = \{u_g, v_g\} \in E$ paarweise verschieden
 2. Es gelte $e \sim f$ und $f \sim g$ **zu Zeigen:** $e \sim g$
 3. Füge Mittelknoten w_e zu der Kante e ein. D.h. w_e ist adjazent zu u_e und v_e und diese beiden Kanten ersetzen Kante e . Füge auch die Mittelknoten w_f und w_g ein.
 4. $e \sim f \Rightarrow \exists$ zwei intern-knotendisjunkte Wege von w_e nach w_f .
 5. Also gibt es kein $w_e - w_f$ -Separator der Grösse 1.
 6. Ebenso gibt es kein $w_f - w_g$ -Separator der Grösse 1 wegen $f \sim g$.
 7. Weder w_e von w_f noch w_f von w_g ist mit einer Separator der Grösse 1 zu trennen.
 8. Also gibt es kein $w_e - w_g$ -Separator der Grösse 1.
 9. Satz von Menger $\Rightarrow$ es gibt zwei intern-knotendisjunkte Pfade von w_e nach w_g .
 10. w_e und w_g liegen auf ein Kreis.
- $\Rightarrow e \sim g$ □

Beweis: Transitivität



Block-Graph

Block-Graph

Sei $G = (V, E)$ ein zusammenhängender Graph. Der Block-Graph von G ist der bipartite Graph $T = (A \uplus B, E_T)$ mit

- $A = \{\text{Artikulationsknoten von } G\}$
- $B = \{\text{Blöcke von } G\}$
- $\forall a \in A, b \in B : \{a, b\} \in E_T \iff a \text{ inzident zu einer Kante in } b$

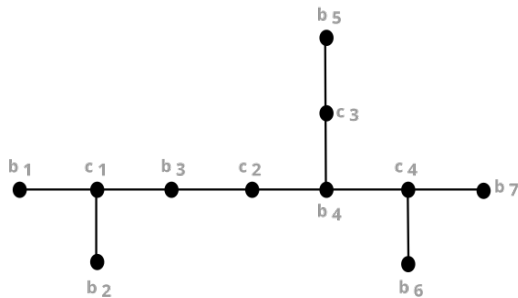
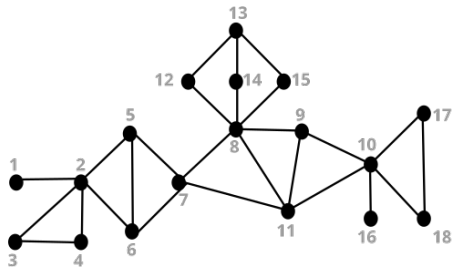
In Wörter: Wir verbinden einen Artikulationsknoten $a \in A$ mit einem Block $b \in B$ genau dann, wenn a inzident zu einer Kante in b ist.

Falls G zusammenhängend ist, dann ist der Block-Graph ein **Baum**.

Man kann die meisten Probleme durch Blöcke in *Teilprobleme* zerlegen!

⁰Engl. Block-Graph: *block-cut-tree*

Block-Graph: Finde die Blöcke



Ein neues DFS

Wie findet man Artikulationsknoten und Brücken effizient?

Modifiziertes DFS

DFS - Zur Erinnerung

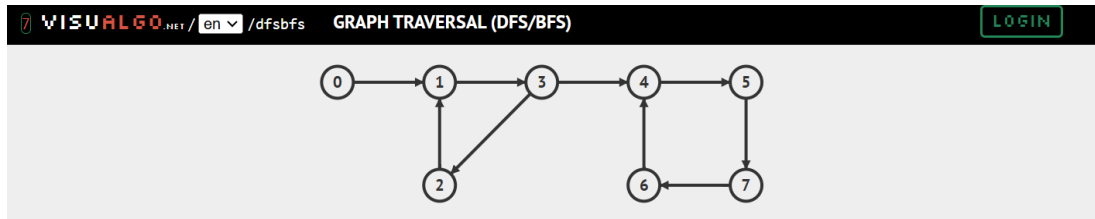
```
1   Markiere v als besucht
2   for all  $\{v, w\} \in E$  do
3       if w ist noch nicht besucht then
4           DFS-VISIT(G, v)
```

DFS-VISIT(G

Falls der Graph zusammenhängend ist, dann besucht der Algorithmus jeden Knoten.

DFS + Reihenfolge

In welcher Reihenfolge besucht DFS die Knoten? Animation:



<https://visualgo.net/en/dfsdfs>

Modifiziertes DFS

- Wir speichern Reihenfolge in einem Array: $dfs[]$.
- Für Startknoten s gilt $dfs[s] = 1$.
- Also bedeutet $dfs[v] = 0$, dass v noch nicht besucht ist.
- Wir speichern die benutzten Kanten auch in einem Array und konstruieren *DFS-Baum*.

Pseudocode

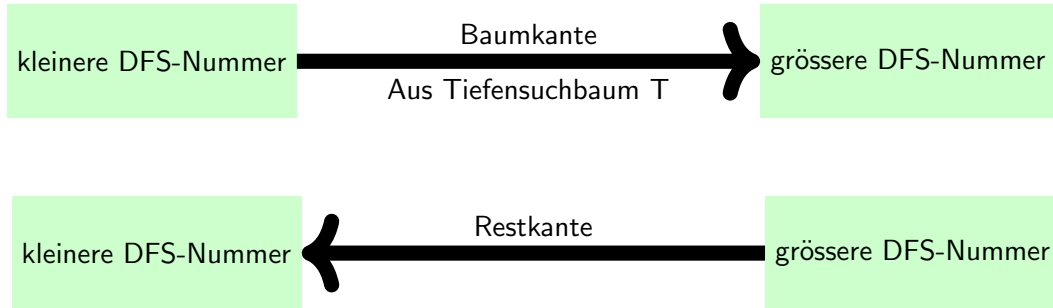
```
1   $\forall v \in V: \text{dfs}[v] \leftarrow 0$       \\keep track of ordering, 0 initialize
2   $\text{num} \leftarrow 0$                 \\keep track of how many vertices visited
3   $T \leftarrow \emptyset$              \\Initialize edge set for Tree edges
4  DFS-VISIT( $G, s$ )
5
```

DFS(G, s) - Rahmenprogramm

```
1   $\text{num} \leftarrow \text{num} + 1$ 
2   $\text{dfs}[v] \leftarrow \text{num}$ 
3  for all  $\{v, w\} \in E$  do
4      if  $\text{dfs}[w] = 0$  then
5           $T \leftarrow T + \{v, w\}$ 
6          DFS-VISIT( $G, s$ )
7
```

DFS-VISIT(G, s)

Kanten neu richten



⁰Engl. *Tree Edge, Back Edge: If the graph is undirected all "Restkanten" are back edges* -wikipedia

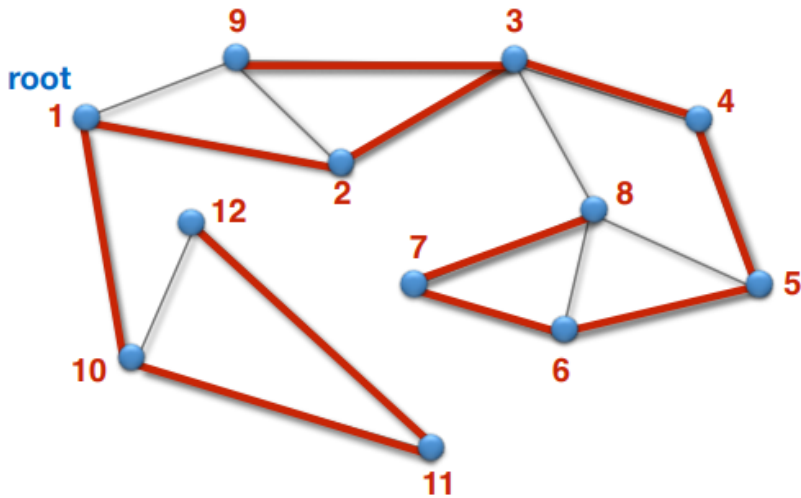
Low Wert

Low-Wert

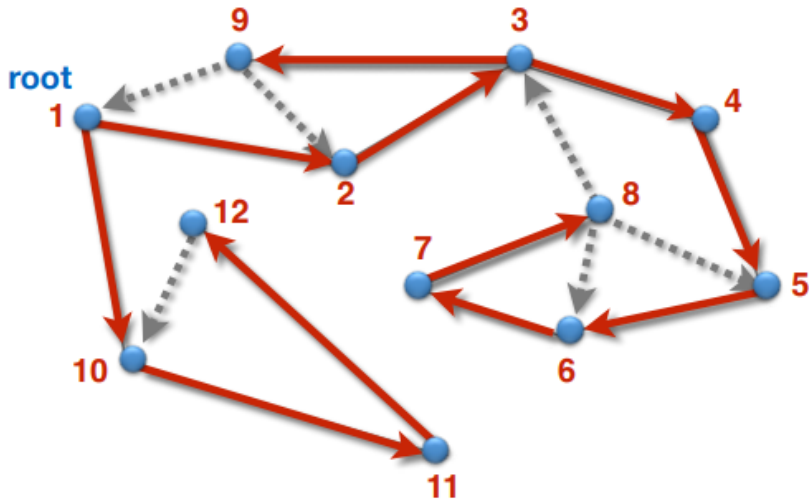
$\text{low}[v] :=$ kleinste dfs-Nummer die man von v aus durch einen Pfad aus (beliebig vielen) Baumkanten und maximal einer Restkante erreichen kann.

- Für alle Knoten $v \in V : \text{low}[v] \leq \text{dfs}[v]$

Zum Beispiel



Zum Beispiel



Wie kann man Artikulationsknoten finden? - 1

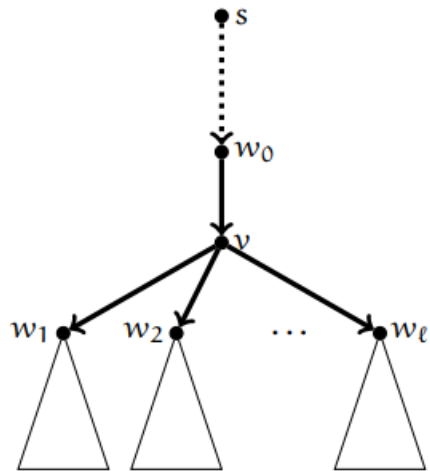
1. Betrachte Knoten $v \neq s$.¹ Falls v Art.knoten besteht $G[V \setminus \{v\}]$ aus mindestens zwei Zusammenhangskomponenten (ZHK): Z_1, Z_2 mit $s \in Z_1$
2. Jeder Pfad von s zu einem Knoten in Z_2 muss v enthalten.
3. Also $1 = \text{dfs}[s] < \text{dfs}[v] < \text{dfs}[w]$ für alle $w \in Z_2$
4. Da $G[Z_2]$ eine ZHK in $G[V \setminus \{v\}]$ ist kann es keine Kanten von einem Knoten $w \in Z_2$ zu einem Knoten in $V \setminus (\{v\} \cup Z_2)$ geben.
5. Also der low-Wert eines jeden Knotens in Z_2 ist mindestens $\text{dfs}[v]$

¹ s = Startknoten

Wie kann man Artikulationsknoten finden? - 2

1. Betrachte Knoten $v \neq s$ der kein Art.knoten ist und seien $w_0, w_1, \dots, w_l$ die Nachbarknoten von $v \in T$.
2. Genau einer dieser Knoten hat dfs-Wert kleiner als v . o.B.d.A sei dies w_0 .
3. Es kann keine Kanten zwischen den Teilbäumen geben, die jeweils an $w_1, \dots, w_l$ hängen. (Wegen DFS)
4. Annahme war v kein Art.knoten. Also muss $G[V \setminus \{v\}]$ zusammenhängend sein.
5. D.h. es muss für jeden Teilbaum eine Restkante zu einem Knoten mit kleinerer dfs-Nummer als v geben.

Wie kann man Artikulationsknoten finden? - 3



Artikulationsknoten finden

Artikulationsknoten finden

$v \in V$ ist Artikulationsknoten

$\iff$

$v = s$ und s hat in T Grad mindestens zwei, oder

$v \neq s$ und es gibt $w \in V$ mit $\{v, w\} \in E(T)$ und $\text{low}[w] \geq \text{dfs}[v]$

Pseudocode - DFS mit low-Wert berechnen

```
1  num  $\leftarrow$  num+1
2  dfs[v]  $\leftarrow$  num
3  low[v]  $\leftarrow$  dfs[v]
4  isArtVert[v]  $\leftarrow$  FALSE
5  for all  $\{v,w\} \in E$  do
6      if dfs[w] = 0 then
7           $T \leftarrow T + \{v,w\}$ 
8          val  $\leftarrow$  DFS-VISIT(G,v)
9          if val  $\geq$  dfs[v] then
10             isArtVert[v]  $\leftarrow$  TRUE
11             low[v]  $\leftarrow$  min{low[v], val}
12         else dfs[w]  $\neq$  0 and  $\{v,w\} \notin T$ 
13             low[v]  $\leftarrow$  min{low[v], dfs[w]}
14  return low[v]
```

DFS-VISIT(G v)

Pseudocode - DFS mit low-Wert berechnen 2

Am Ende kann $\text{isArtVert}[v]$ für den Startknoten s falsch sein. Deswegen müssen wir das im Rahmenprogramm explizit überprüfen:

```
1   $\forall v \in V: \text{dfs}[v] \leftarrow 0$ 
2   $\text{num} \leftarrow 0$ 
3   $T \leftarrow \emptyset$ 
4  DFS-VISIT( $G, s$ )
5  if  $s$  hat in  $T$  Grad mindestens zwei then
6       $\text{isArtVert}[s] \leftarrow \text{TRUE}$ 
7  else
8       $\text{isArtVert}[s] \leftarrow \text{FALSE}$ 
9
```

DFS(G, s)

Satz 1.27.

Für zusammenhängende Graphen $G = (V, E)$, die mit Adjazenslisten gespeichert sind, kann man in Zeit $O(|E|)$ alle Artikulationsknoten und alle Brücken berechnen.

- Eine Kante (v, w) des Tiefensuchbaumes T ist genau dann eine Brücke wenn $\text{low}[w] > \text{dfs}[v]$

Exercises

Let $G = (V, E)$ be a two connected graph.

- (a) Let (u, v, w) be a path in G of length 2. Show that we can extend this path to a cycle, that is, show that G contains a cycle where u, v, w appear as incident vertices.

- (b) Let $e \in E$ be an edge, and $u, v \in V$ be two vertices. Show that there is a path in G from u to v passing through e .
(*Hint:* you may use without proof that in a 2-connected graph, every two edges e, f lie on a common cycle.)

The End

