Wiederholung

$G = (V, E)$ Graph

- $u \in V$ zu ℓ Brücken inzident $\Rightarrow$

  es ex. mindestens ℓ von u versch. Knoten von ungeradem Grad

- $v, w \in V$

  $v \sim w$: $d(v,w) := \min \{ \ell \in \mathbb{N}_0 \mid \exists$ v-w-Pfad der Länge ℓ in $G \}$ Distanz

  $v \not\sim w$: $d(v,w) := \infty$

  - $d(v,w) = 0 \iff v = w$

- Breitensuche / Tiefensuche

  - $w \in V$. Finde zu jedem $v \in V$ mit $v \sim w$
    - $d(v,w)$ und
    - Vorgänger von v auf w-v-Pfad der Länge $d(w,v)$

- Hamiltonkreis: Kreis der Länge n_G

  Eulertour : Tour der Länge m_G

  Eulerzug: Kantenzug der Länge m_G der paarw. versch. Kanten durchläuft.

- Satz: Es sei G zshgd. Es( ) in G genau zwei Knoten u, v von ungeraden Grad, dann ex u-v-Eulerzug.

- Folgerung: Sei G zshgd.

  G besitzt Eulertour $\iff$ $\deg(v)$ gerade $\forall v \in V$.

- Alg. von Fleury: Sei G zshgd. mit $\deg(v)$ gerade $\forall v \in V$.

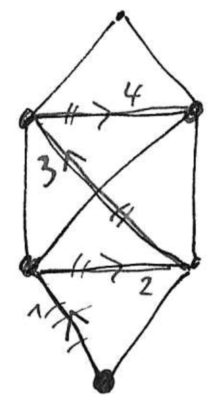
  Starte an beliebigen Knoten v

  Bei jeden Knoten v:

  $\deg(v) = 1$: nehme einzige Kante, die v verlässt

  $\deg(v) > 1$: nehme Nicht-Brücke inzident zu v (ex. immer)

Beispiel



Diese Kante ist jetzt Brücke, darf also nicht genommen werden.

- **Satz**: Sei G zshgd., $n_G \geq 3$.

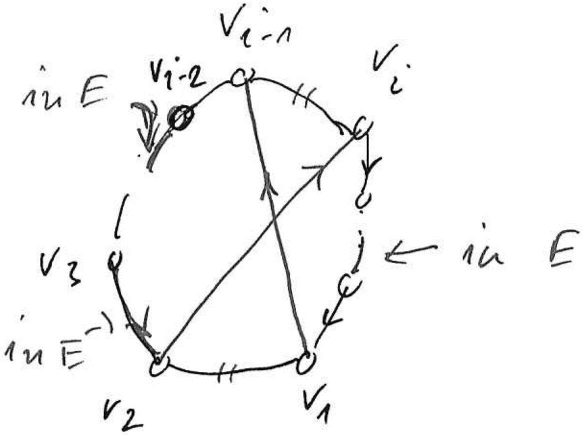
  Für alle $u, v \in V$, $u \neq v$ und $uv \notin E$ ist

  $$\deg(u) + \deg(v) \geq n_G$$

  $\Rightarrow$ G besitzt Hamiltonkreis

Beweis des Satzes $G=(V,E)$, $(v_1,\dots,v_n)$ Permutation von V

$(v_1,v_2,\dots,v_n,v_1)$ Kreis der Länge n in K_n



Sei r die Anzahl der Kanten von $(v_1,\dots,v_n,v_1)$ in E

$r=n$: ✓

$r<n$: O.B.d.A. $v_1v_2 \notin E$

Beh.: $\exists\ i \in \{3,\dots,n\}$ mit $v_1v_{i-1},\ v_2v_i \in E$

Falls Beh. bewiesen: $(v_1,[v_2,v_3,\dots,v_{i-1}],v_i,\dots,v_n,v_1)$

$\updownarrow$ umdrehen

$(v_1,[v_{i-1},v_{i-2},\dots,v_2],v_i,\dots,v_n,v_1)$

Kreis der Länge n in K_n mit $\geq r+1$ Kanten in E.

Bew. der Beh.: Gesucht ist $i \in \{3, \dots, n\}$ mit $v_{i-1} \in \Gamma(v_1)$, $v_i \in \Gamma(v_2)$

$S := \Gamma(v_2)$, $T := \{v_j \mid 2 \leq j \leq n, \; v_{j-1} \in \Gamma(v_1)\}$

Zu zeigen: $S \cap T \neq \emptyset$

$|S| = \deg(v_2)$, $|T| = \deg(v_1)$

$v_1 v_2 \notin E \overset{\text{Voraus.}}{\Longrightarrow} |S| + |T| \geq n$

$v_2 \notin S$, $v_1 \notin T \Rightarrow |S \cup T| < n$

$\Rightarrow |S \cap T| = (|S| + |T|) - |S \cup T| > 0.$ $\blacksquare$

Dijkstras Algorithmus

Definition
Ein *gewichteter Graph* ist ein Tripel $G = (V, E, f)$ mit:
(V, E) ist ein Graph und f eine *Gewichtsfunktion* $f : E \to \mathbb{R}_{\geq 0}$.

Definition
Es sei $G = (V, E, f)$ ein gewichteter Graph.

- Es sei $z = (v_0, \ldots, v_l)$ ein Kantenzug in G.
  $f(z) := \sum_{i=1}^{l} f(v_{i-1}v_i)$ heißt das *Gewicht* von z.
- Für alle $v, w \in V$ mit $v \sim w$ definieren wir die *Distanz* zwischen v und w als

$$d(v, w) := \min\{f(z) \mid z \text{ ist } v\text{-}w\text{-Pfad in } G\} \in \mathbb{R}_{\geq 0}.$$

- Für alle $v, w \in V$ mit $v \not\sim w$ wird $d(v, w) := \infty$ gesetzt.

Dijkstras Algorithmus (Forts.)

DIJKSTRA(Γ, w, f)

initialisiere array $d[1, \ldots, n]$ mit allen Einträgen gleich ∞
initialisiere array $p[1, \ldots, n]$ mit allen Einträgen gleich NIL
initialisiere priority queue Q mit Elementen $1, \ldots, n$ und
allen Prioritäten $= \infty$
$d[w] \leftarrow 0$
INSERT($Q, w, d[w]$)
while Q nicht leer
do $v \leftarrow$ EXTRACTMIN(Q) v aktueller Knoten
    for $u \in \Gamma[v]$
    do if $d[v] + f(uv) < d[u]$
        then $d[u] \leftarrow d[v] + f(uv)$
            $p[u] \leftarrow v$
            INSERT($Q, u, d[u]$)
return d, p

Dijkstras Algorithmus (Forts.)

Kommentare (zum Algorithmus)

- Eingabe:
  - Γ: Adjazenzliste des Graphen $G = (V, E)$ mit $V = \underline{n}$
  - w: Knoten $w \in V$
  - f: Liste der Werte $f(e), e \in E$
- Der `array` $d[1, \ldots, n]$ enthält nach der Terminierung an Position v den Wert $d(w, v)$.
- Der `array` $p[1, \ldots, n]$ enthält nach der Terminierung an Position v einen Knoten u, der auf einem w-v-Pfad der Distanz $d(w, v)$ unmittelbar vor v kommt.

Dijkstras Algorithmus (Forts.)

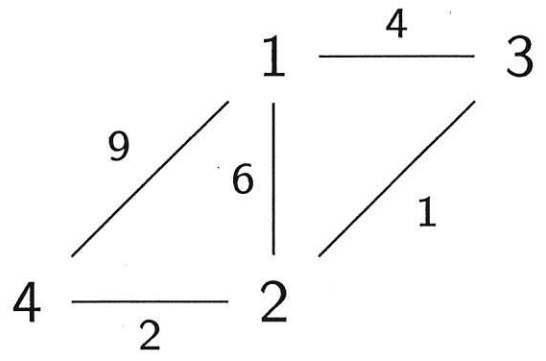
Kommentare (zum Algorithmus), Forts.

- `priority queue` ist eine *Vorrangwarteschlange*, bei der jedem ihrer Element ein *Prioritätswert* zugeordnet ist.
- Der Aufruf $\textsc{Insert}(Q, x, k)$ fügt das Element x in die Warteschlange ein und ordnet x die Priorität $k \geq 0$ zu.

  Falls x bereits in der Warteschlange enthalten ist, wird nur die Priorität neu auf k gesetzt.
- Der Aufruf $\textsc{ExtractMin}(Q)$ entnimmt das Element mit der niedrigsten Priorität.

Dijkstras Algorithmus (Forts.)

Beispiel



d	p	Q	v	$\Gamma(v)$	$d[v] + f(uv) < d[u]$
$[0, \infty, \infty, \infty]$	$[-, -, -, -]$	$\{1, 2, 3, 4\}$	1	$[2, 3, 4]$	$[2, 3, 4]$
$[0, 6, 4, 9]$	$[-, 1, 1, 1]$	$\{2, 3, 4\}$	3	$[1, 2]$	$[2]$
$[0, 5, 4, 9]$	$[-, 3, 1, 1]$	$\{2, 4\}$	2	$[1, 3, 4]$	$[4]$
$[0, 5, 4, 7]$	$[-, 3, 1, 2]$	$\{4\}$	4	$[1, 2]$	$[]$
$[0, 5, 4, 7]$	$[-, 3, 1, 2]$	$\{\}$			

<u>Vorbereitung</u>: $d: [0, \infty, \infty, \infty]$, $p: [-, -, -, -]$, $Q: \{1, 2, 3, 4\}$

Prioritäten: $0, \infty, \infty, \infty$

1. while-Schleife:

$v = 1$, $\Gamma(v) = [2, 3, 4]$ $\quad \underset{0}{d[v]} + \underset{<\infty}{f(uv)} < \underset{\infty}{d[u]} \quad \forall u \in \Gamma(v)$

$d: [0, 6, 4, 9]$ $\quad p: [-, 1, 1, 1]$ $\quad Q: \{2, 3, 4\}$ $\quad 6, 4, 9$

2. while-Schleife:

$v = 3$ $\quad \Gamma(v) = [1, 2]$, $\quad d[v] + f(13) < \underset{0}{d[1]}$ Nein

$\underset{4}{d[v]} + \underset{1}{f(23)} < \underset{6}{d[2]}$ Ja

$d: [0, 5, 4, 9]$, $\quad p: [-, 3, 1, 1]$ $\quad Q: \{2, 4\}$ $\quad 5, 9$

3. while-Schleife

$v = 2$ $\quad \Gamma(v) = [1, 3, 4]$ $\quad \underset{5}{d[v]} + \underset{2}{f(42)} < \underset{9}{d[4]}$ Ja

$d: [0, 5, 4, 7]$, $\quad p: [-, 3, 1, 2]$

4. while-Schleife

$v = 4$, $\quad \Gamma(v) = [1, 2]$ ✓

Beweis der Korrektheit des Dijkstra-Algorithmus:

$v \in V$ besucht, falls $v \notin Q$.

1. Beh: Nach dem Ende jeder while-Schleife gilt $\forall u \in V$:

$d[u] =$ ~~kürz~~ kleinstes Gewicht eines w-u-Pfades, in dem alle Vorgänger von v besucht sind.

Bew.: Klar ~~für~~ nach 0.ter Schleife (oder 1. Schleife)

Nach Induktion ist die Beh. richtig vor der while-Schleife.

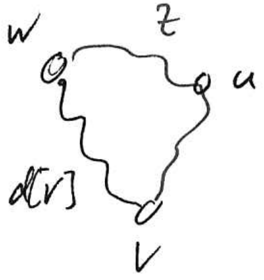
Nach der while-Schleife mit aktuellem Knoten v:

vor der Schleife	nach der Schleife
(a) $d[u] = \infty$	$d[u] < \infty$, $\exists$ genau ein Pfad mit ausschl. besuchten Vorg.
(b) $d[u] < \infty$	
– $d[v] + f(u,v) \geq d[u]$	$d[u]$ wie vorher, kein neuer Pfad
– $d[v] + f(u,v) < d[u]$	$d[u]$ kleiner, neuer Pfad über v, Beh. erfüllt.

2. Beh: Wird v in der while-Schleife aus Q extrahiert, dann ist $d[v] = d(w,v)$ und $d[v]$ wird nicht mehr geändert.

Bew: Vor der while-Schleife: v nicht besucht und

$d[v] = \min\{ d[v'] \mid v' \in V,\ v' \text{ nicht besucht}\}$ $\Leftarrow$



Angenommen, es ex. w-v-Pfad z mit $f[z] < d[v]$

$\overset{\text{1. Beh.}}{\Longrightarrow}$ Nicht alle Vorgänger von v auf z sind besucht.

Sei u der erste (von w aus gesehen) nicht besuchte Knoten auf z

z_1: w-u Anfangsstück von z

z_2: u-v Endstück von z

Aus 1. Beh. folgt: $d[u] \leq f(z_1)$.

$\Rightarrow d[v] > f(z) = f(z_1) + f(z_2) \geq d[u] + f(z_2) \geq d[u]$ ↯ zur Wahl von v

Bäume und Wälder

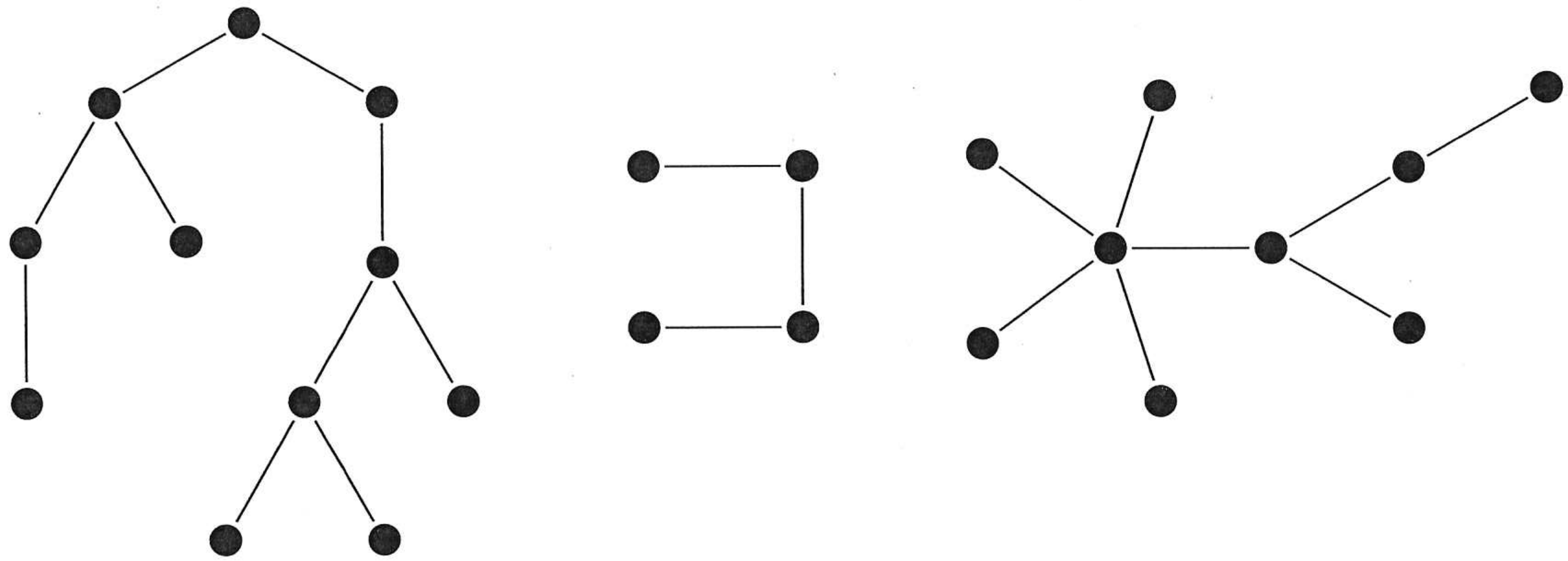
Es sei $G = (V, E)$ ein Graph mit $n_G > 0$.

Definition

- G heißt *kreisfrei* bzw. *Wald*, falls G keine Kreise enthält.
- Ein zusammenhängender Wald heißt *Baum*.
- Die Knoten eines Waldes mit Grad ≤ 1 heißen Blätter.

Wälder und Bäume (Forts.)

Beispiel



Wälder und Bäume (Forts.)

Es sei $G = (V, E)$ ein Graph mit $n_G > 0$.

Bemerkung

(a) ▶ G ist genau dann kreisfrei, wenn jede Kante eine Brücke ist.

(b) ▶ Ist G ein Baum mit $n_G \geq 2$, dann hat G mindestens zwei Blätter.

(c) ▶ Ist G ein Baum mit $n_G \geq 3$, dann hat G höchstens $n_G - 1$ Blätter.

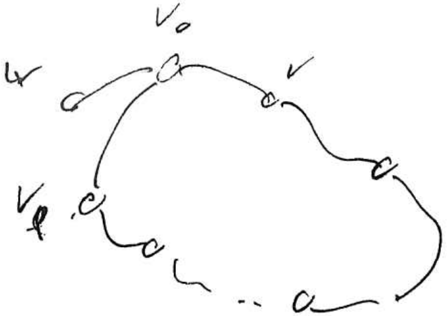
Beweis der Bemerkung:

(a) Sei $e = uv \in E$

e Nicht-Brücke $\iff$ $\exists$ u-v-Kantenzug, der nicht über e fährt

$\iff$ $\exists$ Kreis über e.

(b) Sei $(v_0, v_1, \dots, v_\ell)$ maximaler Pfad (d.h. kann nicht verlängert werden



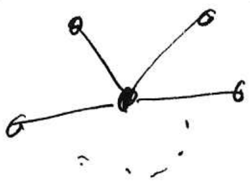
$\deg(v_0) = 1 = \deg(v_\ell)$, sonst gäbe es längeren Pfad oder Kreis.

(c) Andernfalls $\quad n_G \geq \sum_{v \in V} \deg(v) = 2m_G \geq 2(n_G - 1) = 2n_G - 2$

$m_G \geq n_G - r_G$

$\Rightarrow r_G \leq 2$

Beispiel:



$n_G - 1$ Blätter