3. Übung zur Vorlesung

Algorithmentechnik

am

20.11.2008

Die Übungsblätter

- sind zu schwer.
- sind zu lang.
- hab ich mir nicht mal angeschaut.
- halte ich prinzipiell für nicht nützlich.
- gehen am Stoff vorbei.
- würde ich gerne machen, aber ich habe zuviel zu tun.
- ich schau mir einfach die Lösung in der Übung an.
- es gibt Übungsblätter?

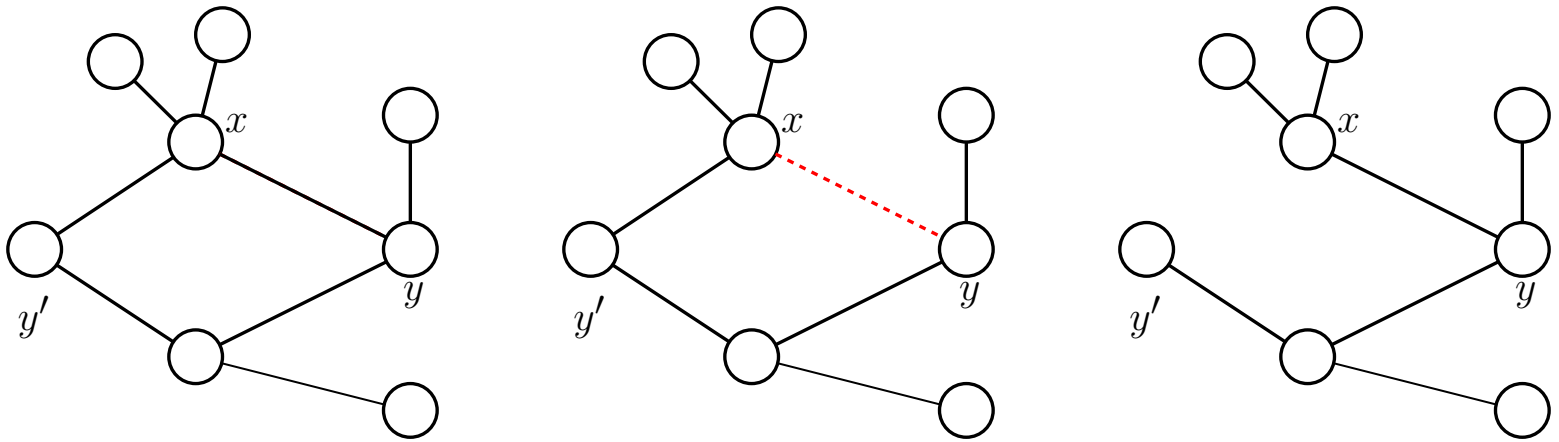
Aufgabe 1: Boruvka-MST

- Sei $G = (V, E)$ ein einfacher, ungerichteter, zusammenhängender Graph mit **injektiver** Kostenfunktion $w : E \to \mathbb{N}^+$.
- Wir schreiben (u, v) für eine ungerichtete Kante $\{u, v\}$.
- Wir bezeichnen eine Kante (u, v) als *lokal minimal* wenn sie für x oder y unter allen Kanten, die mit diesem Knoten inzidieren, das kleinste Gewicht hat.

Teilaufgabe (a).
Zeigen sie: Jede lokal minimale Kante gehört zu allen MST von G.

Zeigen sie: Jede lokal minimale Kante gehört zu allen MST von G.

- Sei (x, y) eine lokal minimale Kante (minimal für x)
- Sei T' MST von G der (x, y) nicht enthält

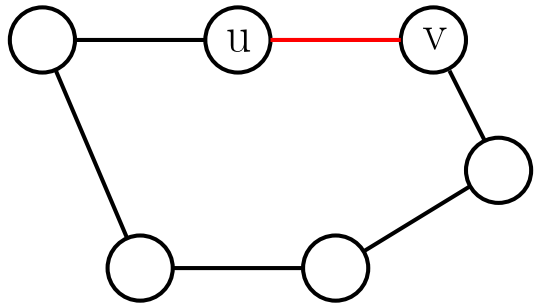


- Dann gibt es in T' eine andere Kante (x, y'), die auf dem Pfad in T' von x nach y liegt.
- entferne (x, y') aus T' füge (x, y) hinzu
- $\Rightarrow$ neuer MST T mit geringerem Gewicht als T'
- $\Rightarrow$ Widerspruch zu T' ist MST.

Folgende Eigenschaften sind zu zeigen:

- $w(T) < w(T')$, denn $w(T) = w(T') - w(x, y') + w(x, y)$, (x, y) ist lokal minimal und Gewichtsfunktion ist injektiv
- T spannt G auf. T' spannt G auf. Für Pfad in T' zwischen zwei beliebigen Knoten v_1, v_2 über (x, y') ersetze die Kante (x, y') durch den Pfad über (x, y) nach y'.
- T ist ein Baum. Da T' Baum ist, müsste ein Zyklus in T die Kante (x, y) enthalten. Also müsste in $T \setminus (x, y) = T' \setminus (x, y')$ einen Pfad von x nach y existieren. Der einzige x-y-Pfad in T' beinhaltet allerdings (x, y').

Teilaufgabe (b). Die Menge aller lokal minimalen Kanten eines Graphen ist ein Wald (zyklenfreier Graph).

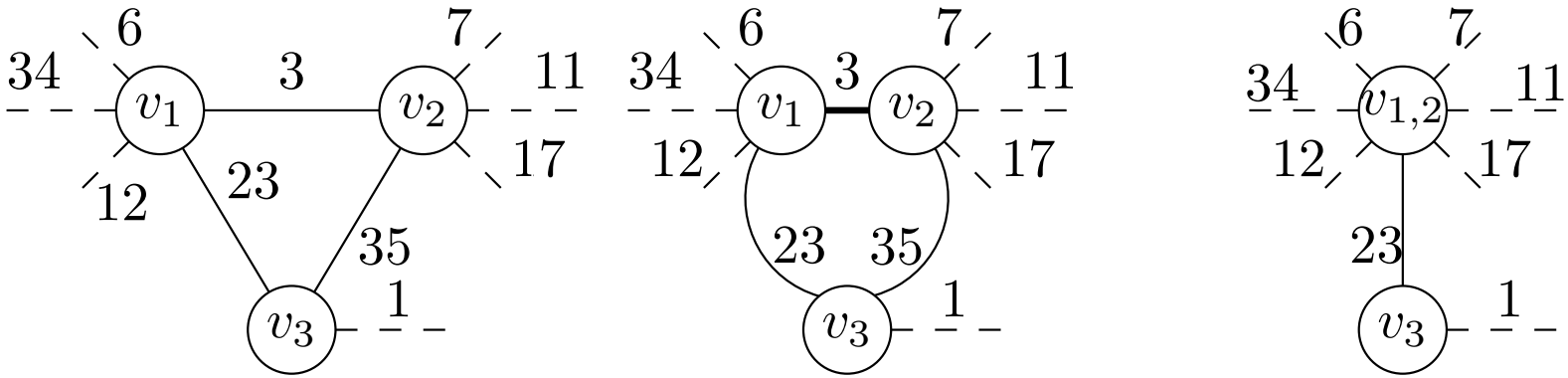


- Angenommen es gäbe in der Menge der lokal minimalen Kanten E_{lm} einen Zyklus.
- Betrachte darauf die Kante größten Gewichts (u, v).
- Diese ist eindeutig, da die Gewichtsfunktion injektiv ist.
- (u, v) kann nun aber keine lokal minimale Kante sein, denn für u und v gibt es bereits auf dem Zyklus je eine anliegende Kante niedrigeren Gewichtes. Widerspruch.

Eine Boruvka-Phase ist ein Algorithmus, der Folgendes leistet:

Schritt (1) Er bestimmt die Menge $L(G)$ aller lokal minimalen Kanten von G.

Schritt (2) Er berechnet den kontrahierten Graphen $G/L(G)$, wie in der Abbildung unten illustriert. Falls Mehrfachkanten entstünden, wird nur die Kante mit geringsten Gewicht behalten, und die anderen entfernt.



Teilaufgabe (c).

- Implementieren sie in Pseudocode die Boruvka-Phase mit einer Worst-case-Laufzeit von $O(m)$ Vergleichsoperationen.
- Begründen Sie die Laufzeit.
- Ihnen steht eine Funktion LöscheTeureMehrfachkanten(G) zur Verfügung, die (im Worst-case) mit $O(m)$ Vergleichsoperationen auskommt.

Algorithmus 1 : BORUVKA-PHASE

Eingabe : Graph $G(V_G, E_G)$

Ausgabe : Kantenmenge $L(G)$, Graph $B(G)(V_B, E_B)$

1 $L(G) \leftarrow \emptyset$

2 $V_B \leftarrow V_G$

3 $E_B \leftarrow E_G$

4 **Für** $v \in V_G$

5 Bestimme lokal minimale Kante e von v

6 $L(G) \leftarrow L(G) \cup \{e\}$

7 **Für** *jede Zusammenhangskomponente* $C = \{v_1, \ldots, v_k\}$ *in dem von* $L(G)$ *induzierten Subgraphen von* G

8 füge Knoten v_C in V_B ein

9 lösche alle in $C \times C$ enthaltenen Kanten aus E_B

10 ersetze in E_B alle Kanten der Form (v_i, w) mit $w \notin C$ durch (v_C, w)

11 LÖSCHETEUEREMEHRFACHKANTEN($B(G)$)

Teilaufgabe (d).
Zeigen sie, dass durch eine Boruvka-Phase die Anzahl der Knoten um mindestens die Hälfte reduziert wird.

- Für jede lokal minimale Kante im Graphen wird durch das Verschmelzen die Anzahl der Knoten um 1 verringert.
- Jeder Knoten besitzt eine lokal minimale Kante.
- Da jede Kante nur zwei Endpunkte hat, kann sie für höchstens zwei Knoten lokal minimal sein.
- Also gibt es mindestens $n/2$ lokal minimale Kanten, die in einer Boruvka-Phase kontrahiert werden.

Teilaufgabe (e).

- Nutzen sie Teilaufgabe (c) um einen MST mit $O(m \log n)$ Vergleichsoperationen im Worst-case zu bestimmen (Pseudocode).
- Begründen sie die Worst-case-Laufzeit von $O(m \log n)$.

Algorithmus 2 : BORUVKA-MST

Eingabe : Graph $G(V, E)$
Ausgabe : MST von G

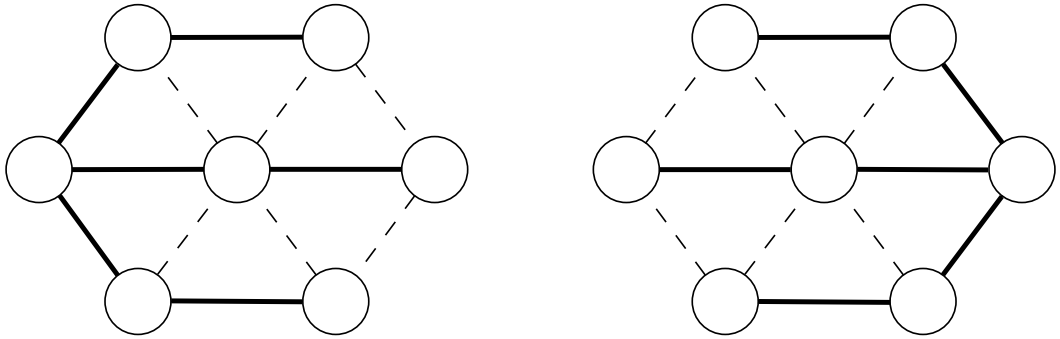
```
1 T ← ∅
2 solange |G(V)| > 1 tue
3 |   L(G), G ← Boruvka-Phase(G)
4 |   T ← T ∪ L(G)
5 return L(G)
```

Aufgabe 2: Schnitte und MSTs

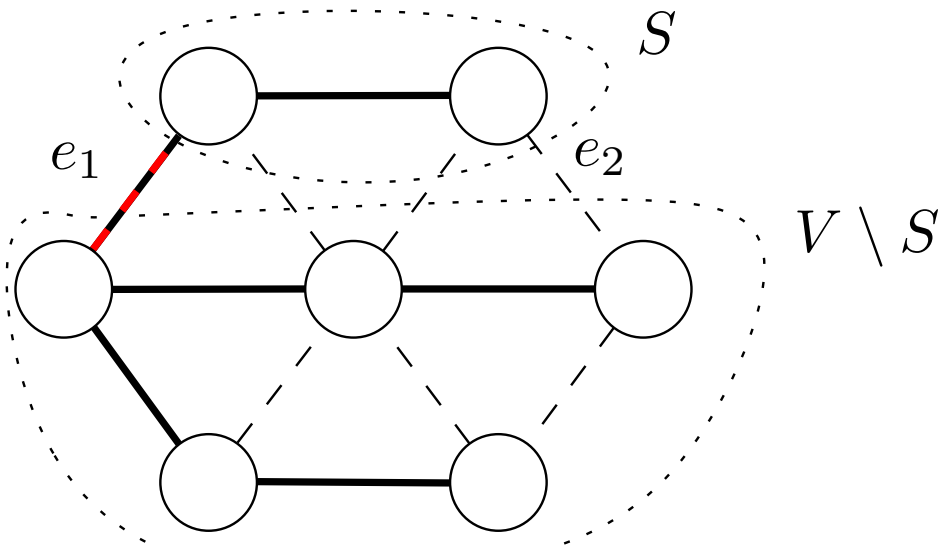
Teilaufgabe (a). Zeigen Sie:

Wenn in einem Graph G für jeden Schnitt die den Schnitt kreuzende Kante minimalen Gewichts eindeutig ist, dann hat G einen eindeutigen aufspannenden Baum minimalen Gewichts.

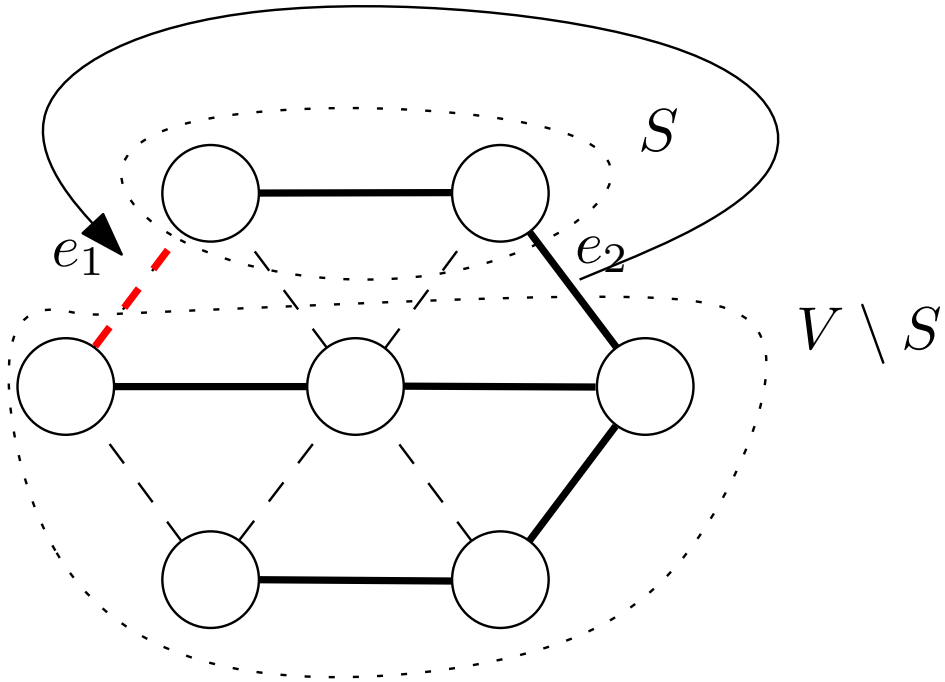
Wir nehmen an, dass die Voraussetzung erfüllt ist, aber G zwei verschiedene MSTs T_1 und T_2 hat.



- Sei $e_1 \in E(T_1) \setminus E(T_2)$ eine Kante von T_1, die nicht in T_2 ist.
- Diese partitioniert T_1 in zwei disjunkte Teilbäume.
- Wir fassen diese als Schnitt $(S, V \setminus S)$ auf.



- Unter allen Kanten, die den Schnitt kreuzen, hat e_1 minimales Gewicht (sonst wäre T_1 nicht minimal).
- Andererseits induziert e_1 in T_2 einen Kreis.
- Mindestens eine weitere der Kanten dieses Kreises kreuzt den Schnitt. Diese Kante sei e_2.



- Wegen $c(e_1) < c(e_2)$ ist $T_2 \setminus \{e_2\} \cup \{e_1\}$ ein Baum mit echt kleinerem Gewicht als T_2, ein Widerspruch.

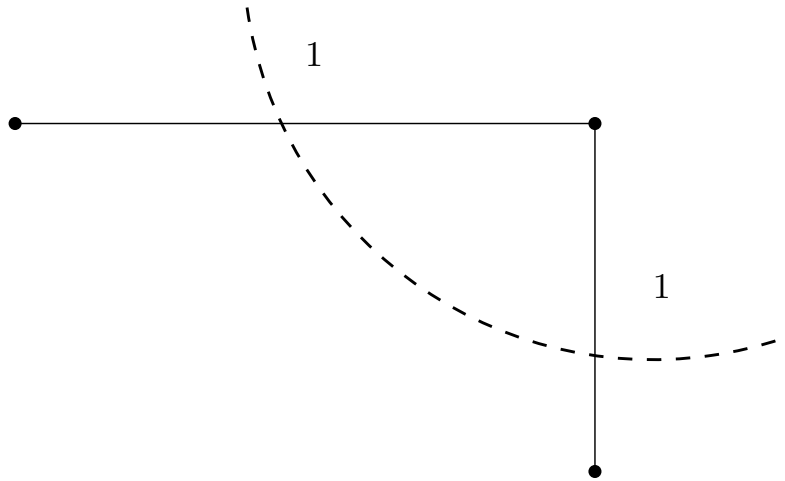
(a) Wenn in einem Graph G für jeden Schnitt die den Schnitt kreuzende Kante minimalen Gewichts eindeutig ist, dann hat G einen eindeutigen aufspannenden Baum minimalen Gewichts.

Teilaufgabe (b). Gilt die Umkehrung von (a) ?

(a) Wenn in einem Graph G für jeden Schnitt die den Schnitt kreuzende Kante minimalen Gewichts eindeutig ist, dann hat G einen eindeutigen aufspannenden Baum minimalen Gewichts.

Teilaufgabe (b). Gilt die Umkehrung von (a) ?

Nein, Gegenbeispiel:



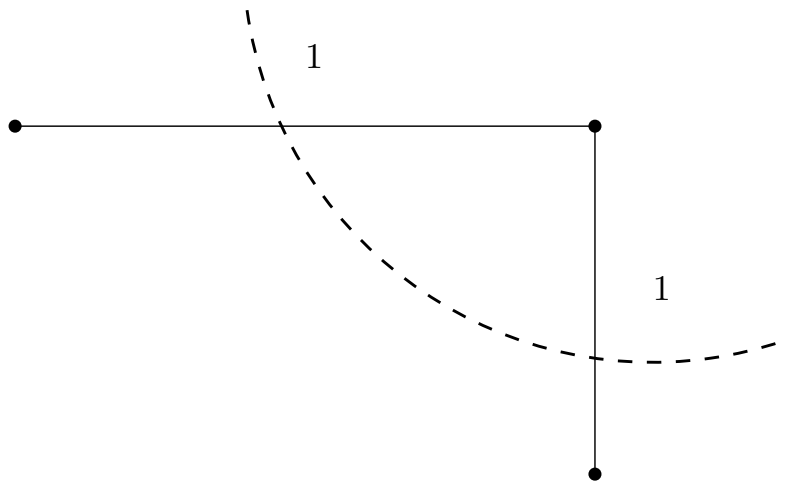
Teilaufgabe (c). Zeigen oder widerlegen Sie: Wenn in einem Graph G mit reellen Kantengewichten alle Kanten paarweise verschiedene Gewichte haben, dann hat G einen eindeutigen aufspannenden Baum minimalen Gewichts.

Folgt aus (a): Wenn alle Gewichte verschieden sind, dann gibt es unter den Kanten, die einen Schnitt kreuzen, stets genau eine mit minimalem Gewicht.

(c) Wenn in einem Graph G mit reellen Kantengewichten alle Kanten paarweise verschiedene Gewichte haben, dann hat G einen eindeutigen aufspannenden Baum minimalen Gewichts.

Teilaufgabe (d). Zeigen oder widerlegen Sie die Umkehrung der Aussage in (c).

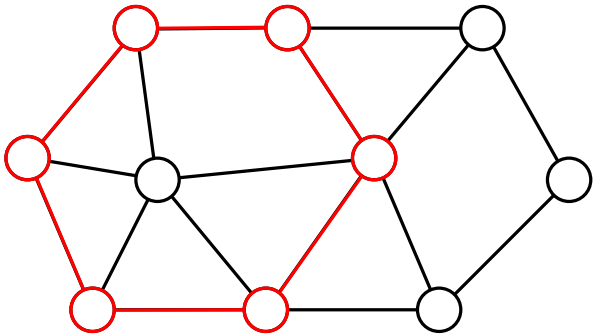
Die Umkehrung gilt nicht, Gegenbeispiel:



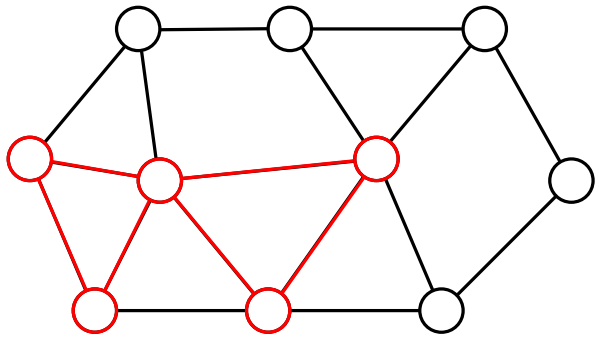
Aufgabe 3: Vier Äquivalente Aussagen

Zur Erinnerung:

- Ein Zyklus ist eine Knotenfolge, bei der Anfangs- und Endknoten übereinstimmen und zwischen je zwei aufeinanderfolgenden Knoten eine Kante existiert.
- Ein Kreis ist ein Zyklus, bei dem keine Knoten außer dem Anfangsknoten mehrmals vorkommen.
- In jedem Zyklus ist ein Kreis enthalten.



Kreis



Zyklus

Zeigen Sie, dass die folgenden Aussagen für einen ungerichteten Graphen $G = (V, E)$ äquivalent sind.

(a) G ist zusammenhängend und enthält keine Kreise.

(b) G ist maximal kreisfrei, d.h. G enthält keinen Kreis und für je zwei Knoten $u, v \in V$ mit $\{u, v\} \notin E$ enthält der Graph $G' = (V, E \cup \{\{u, v\}\})$ einen Kreis.

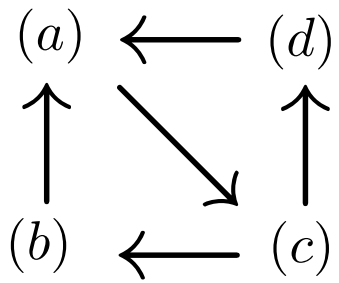
(c) Zwischen je zwei Knoten in V gibt es genau einen Weg in G.

(d) G ist minimal zusammenhängend, d.h. G ist zusammenhängend, und für jede Kante $e \in E$ ist der Graph $G' = (V, E \setminus \{e\})$ nicht zusammenhängend.

Vorbemerkung

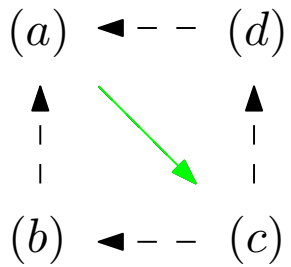
- Falls G nur einen Knoten enthält, dann sind die Aussagen trivialerweise äquivalent.
- Also bestehe der Graph G im folgenden aus mindestens zwei Knoten.

Folgende Implikationen induzieren die Äquivalenz der vier Aussagen:



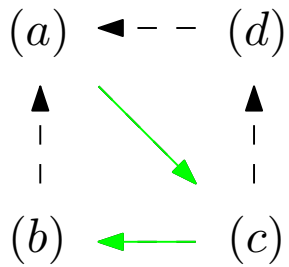
$(a \Rightarrow c)$: (zusammenhängend und kreisfrei $\Rightarrow$ zwischen je zwei Knoten gibt es genau einen Weg)

- Der Graph G enthalte zwei Knoten v und w, zwischen denen 0 oder 2 Pfade existieren.
- Falls es keinen Weg gibt, dann ist G nicht zusammenhängend.
- Falls es zwei verschiedene Pfade W_1 und W_2 gibt, dann ergibt die folgende Knotenfolge einen Zyklus: v, W_1, w, W_2, v.
- Daraus folgt, dass G nicht kreisfrei ist.



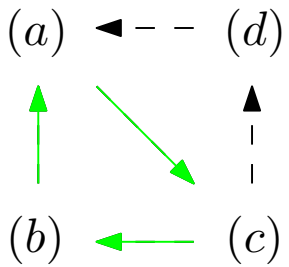
$(c \Rightarrow b)$: (genau ein Weg zwischen je zwei Knoten $\Rightarrow$ maximal kreisfrei)

- Falls G einen Kreis enthält, dann gibt es mindestens zwei Knoten, zwischen denen es mehr als einen Pfad gibt.
- Betrachte nun zwei beliebige Knoten v und w, so dass $\{w, v\} \notin E$.
- Zwischen v und w gibt es genau einen Pfad W in G.
- In $G' := (V, E \cup \{\{w, v\}\})$ gäbe es dann zwei verschiedene Pfade von v nach w.
- Daraus folgt, dass G einen Zyklus (v, W, w, v) und somit einen Kreis enthalten müsste.



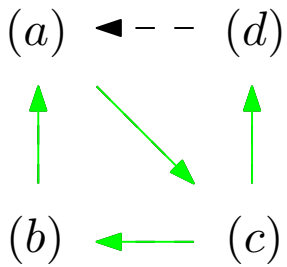
$(b \Rightarrow a)$: (maximal kreisfrei $\Rightarrow$ zusammenhängend und kreisfrei)

- Wäre G nicht zusammenhängend, so würden mindestens zwei Zusammenhangskomponenten Z_1 und Z_2 existieren.
- Wähle zwei Knoten $b_1 \in Z_1$ und $b_2 \in Z_2$ und verbinde sie durch eine Kante e.
- Diese Kante induziert keinen Kreis. Also wäre G nicht maximal kreisfrei.
- Somit ist G zusammenhängend und enthält außerdem keine Kreise.



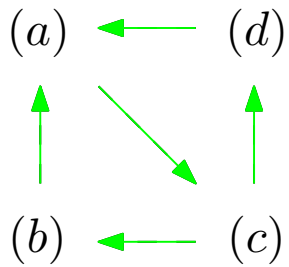
$(c \Rightarrow d)$: (genau ein Weg zwischen je zwei Knoten $\Rightarrow$ minimal zusammenhängend)

- G ist offensichtlich zusammenhängend.
- Betrachte die Endknoten v und w einer beliebigen Kante $e \in E$.
- Diese Kante ist der einzige Pfad von v nach w.
- Durch Entfernen von e würde kein Weg mehr zwischen v und w existieren, d.h. diese Knoten würden zwei verschiedenen Zusammenhangskomponenten angehören.
- Somit wäre $G' = (V, E \setminus \{e\})$ nicht zusammenhängend.
- Da e beliebig gewählt wurde folgt die Behauptung.



$(d \Rightarrow a)$: (minimal zusammenhängend $\Rightarrow$ zusammenhängend und kreisfrei)

- Wir zeigen durch einen Widerspruchsbeweis, dass G keine Kreise enthält.
- Würde G einen Kreis enthalten, dann könnten wir eine Kante aus diesem Kreis entfernen und der resultierende Graph bliebe immer noch zusammenhängend.
- Somit ist G zusammenhängend und kreisfrei.



Wiederholung Matroide

Unabhängigkeitssysteme

Ein Tupel $(M, \mathcal{U})$ wobei $\mathcal{U} \subset 2^M$ ein Mengensystem über einer endlichen Menge M ist, heißt Unabhängigkeitssystem, wenn

- $\emptyset \in \mathcal{U}$ und
- $I_1 \in \mathcal{U}, I_2 \subseteq I_1 \Rightarrow I_2 \in \mathcal{U}$.

Die Mengen $I \subseteq M$ mit $I \in \mathcal{U}$ werden unabhängig, alle anderen Mengen $I \subseteq M$ abhängig genannt.

Basen von Unabhängigkeitssystemen

Sei $(M,\mathcal{U})$ ein Unabhängigkeitssystem.

- Für $F \subseteq M$ ist jede unabhängige Menge $U \in \mathcal{U}$, $U \subseteq F$ die bezüglich $\subseteq$ maximal ist eine Basis von F.
- Eine Basis von M wird Basis des Unabhängigkeitssystems $(M,\mathcal{U})$ genannt.
- Die Menge aller Basen von $(M,\mathcal{U})$ heißt *Basissystem von* $(M,\mathcal{U})$

Optimierungsprobleme über Unabhängigkeitssystemen

- Betrachte ein Unabhängigkeitssystem $(M, \mathcal{U})$ mit einer Gewichtsfuntion w auf M.
- Das Problem eine unabhängige Menge $U^* \in \mathcal{U}$ zu finden, so dass $w(U^*)$ maximal ist, heißt *Optimierungsproblem über dem Unabhängigkeitssystem* $(M, \mathcal{U})$.
- Sei $\mathcal{B}$ Basissystem. Dann heißt das Problem eine Basis $B^* \in \mathcal{B}$ zu finden, so dass $w(B^*)$ minimal ist ein *Optimierungsproblem über dem Basissystem* $\mathcal{B}$.

Beispiel

- Das MST-Problem ist ein Minimierungsproblem genauer ein Optimierungsproblem über dem Basissystem der „aufspannenden Bäume“.

Algorithmus 3 : Greedy–Methode für ein Optimierungsproblem Π über $(M, \mathcal{U})$, $|M| = n$

Sortiere M aufsteigend (absteigend), falls Π Minimierungsproblem über Basissystem (Maximierungsproblem über Unabhängigkeitssystem), die Sortierung sei $\ell_1, \ell_2, \ldots, \ell_n$.

$I^* \leftarrow \emptyset$

Für $i = 1, \ldots, n$

  Wenn $I^* \cup \{\ell_i\} \in \mathcal{U}$

    $I^* \leftarrow I^* \cup \{\ell_i\}$

Laufzeit der Greedy-Methode:

- Aufwand mindestens Sortieren der Elemente
- Laufzeit hängt davon ab, wie schnell die Frage

  Ist I^* zusammen mit dem neuen Element unabhängig?

  beantwortet werden kann.
- Dies hängt vom Unabhängigkeitssystem ab.

Frage:

- Wann liefert die Greedy-Methode ein optimales Ergebnis?

Matroid

- Ein Unabhängigkeitssystem $(M, \mathcal{U})$ heißt Matroid, wenn für alle $I, J \in \mathcal{U}$ mit $|I| < |J|$, ein $e \in J \setminus I$ existiert, so dass $I \cup \{e\} \in \mathcal{U}$.
- Bemerkung: Äquivalent kann anstatt $|I| < |J|$ auch $|I| + 1 = |J|$ gefordert werden.

Optimalität für Optimierungsproblem über Unabhängigkeitssystemen (Maximierungsprobleme)

Für ein Unabhängigkeitssystem $(M, \mathcal{U})$ sind folgende Aussagen äquivalent:

(*a*) Eine Greedy-Methode liefert eine Optimallösung für das Optimierungsproblem $\Pi = \max\{w(U) \ : \ U \in \mathcal{U}\}$ über dem Unabhängigkeitssystem $(M, \mathcal{U})$ bei beliebiger Gewichtsfunktion w über M.

(*b*) $(M, \mathcal{U})$ ist ein Matroid.

(*c*) Für eine beliebige Menge $F \subseteq M$ und beliebige inklusionsmaximale unabhängige Mengen $I_1, I_2 \subseteq F$ gilt: $|I_1| = |I_2|$.

Optimalität für Optimierungsproblem über Basissystemen (Minimierungsprobleme)

Sei $(M,\mathcal{U})$ ein Unabhängigkeitssystem und $\mathcal{B}$ das Basissystem von $(M,\mathcal{U})$. Dann sind äquivalent:

(a) Die Greedy-Methode für Basissysteme liefert eine Optimallösung für das Problem $\Pi = \min\{w(B) : \ B \in \mathcal{B}\}$ für beliebige Gewichtsfunktion w über M.

(b) $(M,\mathcal{U})$ ist ein Matroid.

Aufgabe 4: Einmaschinenscheduling

- Eine Reihe von Aufträgen $A_1, \ldots, A_n$ müssen von einer einzigen Maschine abgearbeitet werden, es kann also zu jedem Zeitpunkt höchstens ein Auftrag bearbeitet werden.
- Alle Aufträge benötigen die gleiche Bearbeitungszeit.
- Jeder Auftrag A_i hat eine Deadline $D_i \in \mathbb{R}$ bis zu der er fertig sein muss.

Sei o.B.d.A.

- die Bearbeitungszeit eines Auftrages auf 1 normiert
- $D_1 \leq D_2 \leq \cdots \leq D_n$.

Teilaufgabe a): Zeigen Sie, dass die Menge aller Teilmengen von Aufträgen, die rechtzeitig bearbeitet werden können, ein Matroid ist.

- Eine Menge von Aufträgen, die rechtzeitig bearbeitet werden können, heiße *zulässig*.
- Offensichtlich: Die Bearbeitungszeit für j Aufträge beträgt j.
- Eine Teilmenge $\{A_{i_1}, \ldots, A_{i_k}\}$ von k paarweise verschiedenen Aufträgen $(i_1 < \cdots < i_k \in \{1, \ldots, n\})$ ist genau dann zulässig, wenn gilt: $\forall j \leq k : D_{i_j} \geq j$

Die Menge aller Teilmengen von Aufträgen, die rechtzeitig bearbeitet werden können ist Unabhängigkeitssystem, denn

- eine leere Menge von Aufträgen kann stets rechtzeitig bearbeitet werden.
- eine Teilmenge einer zulässigen Menge von Aufträgen kann ebenfalls rechtzeitig bearbeitet werden.

Austauscheigenschaft:

- Seien $\mathcal{I} := \{A_{i_1}, \ldots, A_{i_k}\}$ und $\mathcal{J} := \{A_{j_1}, \ldots, A_{j_\ell}\}$ zwei zulässige Mengen mit $k < \ell$ und $i_1 < i_2 < \ldots < i_k$ sowie $j_1 < j_2 < \ldots < j_l$.
- Seien k' bzw. ℓ' die minimalen Indizes mit $i_{k'+1} = j_{\ell'+1}$, $i_{k'+2} = j_{\ell'+2}$, ..., $i_k = j_\ell$,

$$\begin{array}{ccccccccc} A_{i_1} & \ldots & A_{i_{k'}} & & A_{i_{k'+1}} & & A_{i_{k'+2}} & \ldots & A_{i_k} \\ & & \neq & & \neq & & = & & = \\ A_{j_1} & \ldots & \ldots & & A_{j_{l'}} & & A_{j_{l'+1}} & \ldots & A_{j_l} \end{array}$$

Behauptung: $\mathcal{I}' := \{A_{i_1}, \ldots, A_{i_{k'}}, A_{j_{\ell'}}, A_{i_{k'+1}}, \ldots, A_{i_k}\}$ ist zulässig.

$$\begin{array}{ccccccccc} A_{i_1} & \ldots & A_{i_{k'}} & A_{i_{k'+1}} & A_{i_{k'+2}} & \ldots & A_{i_k} \\ & & \neq & \neq & = & & = & & \\ A_{j_1} & \ldots & \ldots & A_{j_{l'}} & A_{j_{l'+1}} & A_{j_{l'+2}} & \ldots & A_{j_l} \end{array}$$

Idee:

- Falls $A_{i_k} \neq A_{j_\ell}$ dann kann A_{j_ℓ} an $\mathcal{I}$ angehängt werden, da $|\mathcal{J}| > |\mathcal{I}|$ und $\mathcal{J}$ zulässig.
- Falls $A_{i_k} = A_{j_\ell}$ dann laufe in $\mathcal{I}$ und $\mathcal{J}$ solange parallel rückwärts, bis der erste Unterschied $A_{i_{k'}} \neq A_{j_{\ell'}}$ auftritt. Dann, füge $A_{j_{\ell'}}$ nach $A_{i_{k'}}$ in $\mathcal{I}$ ein.
- Da die Elemente $A_{j_{\ell'}}$ bis A_{j_ℓ} in $\mathcal{J}$ zulässig sind, sind sie es auch in $\mathcal{I}$.

Etwas formaler:

- Für $\nu \leq k'$ gilt $D_{i_\nu} \geq \nu$ da $\mathcal{I}$ zulässig ist.
- Da $\ell > k$ ist, gilt auch $\ell' > k'$, also $\ell' \geq k' + 1$.
- Daraus folgt $D_{i_{\ell'}} \geq \ell' \geq k' + 1$, da $\mathcal{J}$ zulässig ist.
- Außerdem gilt für $1 \leq \mu \leq k - k'$, dass $D_{i_{k'+\mu}} = D_{j_{\ell'+\mu}} \geq \ell' + \mu \geq k' + 1 + \mu$ ist.
- Also ist $\mathcal{I}'$ zulässig.

Teilaufgabe b:)

Wird ein Auftrag nicht rechtzeitig fertig, so muss eine Strafe P_i bezahlt werden, deren Höhe vom jeweiligen Auftrag abhängt.

In welcher Reihenfolge sollten die Aufträge abgearbeitet werden, damit die Gesamtstrafe minimiert wird?

- **Beobachtung 1:** Die Gesamtstrafe hängt nur von der Menge der nicht bearbeiteten Aufträge ab.
- **Schritt 1:** Finde zulässige Menge S für die $\sum_{s \in S} P_s$ maximal ist

- **Beobachtung 2:** Wenn S eine zulässige Menge ist, dann findet man eine "zulässige" Reihenfolge indem man die Elemente in nicht-aufsteigender Deadline bearbeitet
- **Schritt 2:** Bearbeite die in Schritt 1 gefundene Menge mit nicht-aufsteigender Deadline

Greedy-Ansatz für Einmaschinenscheduling:

1. Sortiere die Aufträge in nicht-aufsteigender Reihenfolge ihrer Strafen.
2. Setze $S := \emptyset$.
3. Für jeden Auftrag A in obiger Reihenfolge
4. Wenn $S \cup \{A\}$ zulässig ist
5. Setze $S := S \cup \{A\}$.
6. Bearbeite die Aufträge in S in nicht-absteigender Reihenfolge ihrer Deadlines