Grundlagen Theoretischer Informatik 3

SoSe 2010 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Grundlagen Theoretischer Informatik 3 Gesamtübersicht

- Organisatorisches; Einführung
- Algorithmenanalyse: Komplexität und Korrektheit
- Zur Behandlung NP-schwerer Probleme: Ausblicke
- Randomisierte Algorithmen und ihre Analyse

Zum Entwurf von Approximationsalgorithmen

—Entwurfstechniken

—Beweistechniken für behauptete Schranken

—Das Finden von bösen Beispielen

—Allgemeines Ziel: Polynomzeit-Approximation von konstanter Güte (APX)

Entwurfstechniken

—Greedy
—Lokale Suche
—problemspezifische Techniken
—und vieles mehr

Beweistechniken für behauptete Schranken

—Im Allgemeinen nicht-triviale Aufgabe

—hier steckt die Hauptschwierigkeit bei approximativen Algorithmen

—oft sind allerdings recht triviale Schranken dienlich:

Maximiert man beispielsweise die Anzahl von Kanten mit gewisser Eigenschaft, so ist das Optimum nicht größer als die Anzahl aller Kanten.

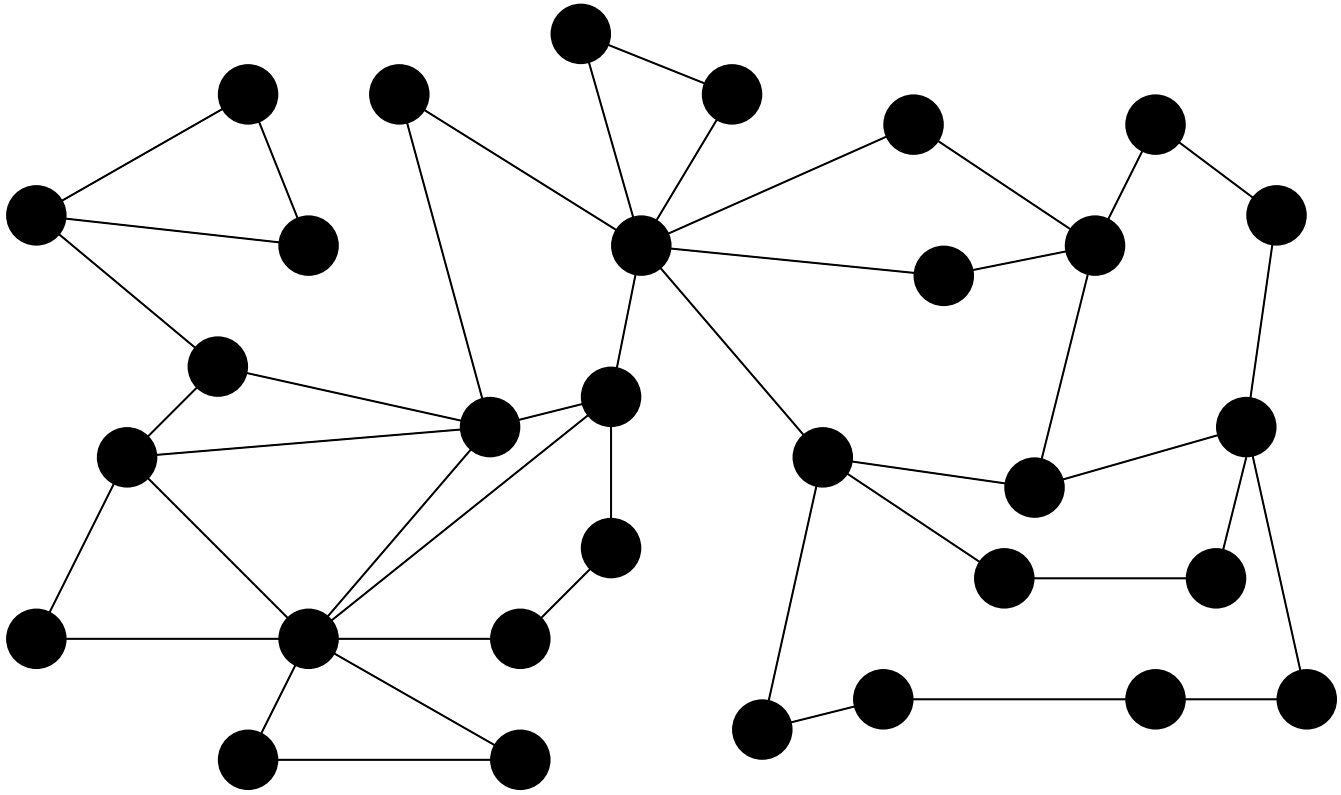
Bei Minimierungsproblemen nutzt man oft lokale Eigenschaften.

Das Knotenüberdeckungsproblem: Ist Greedy gut? GreedyVC

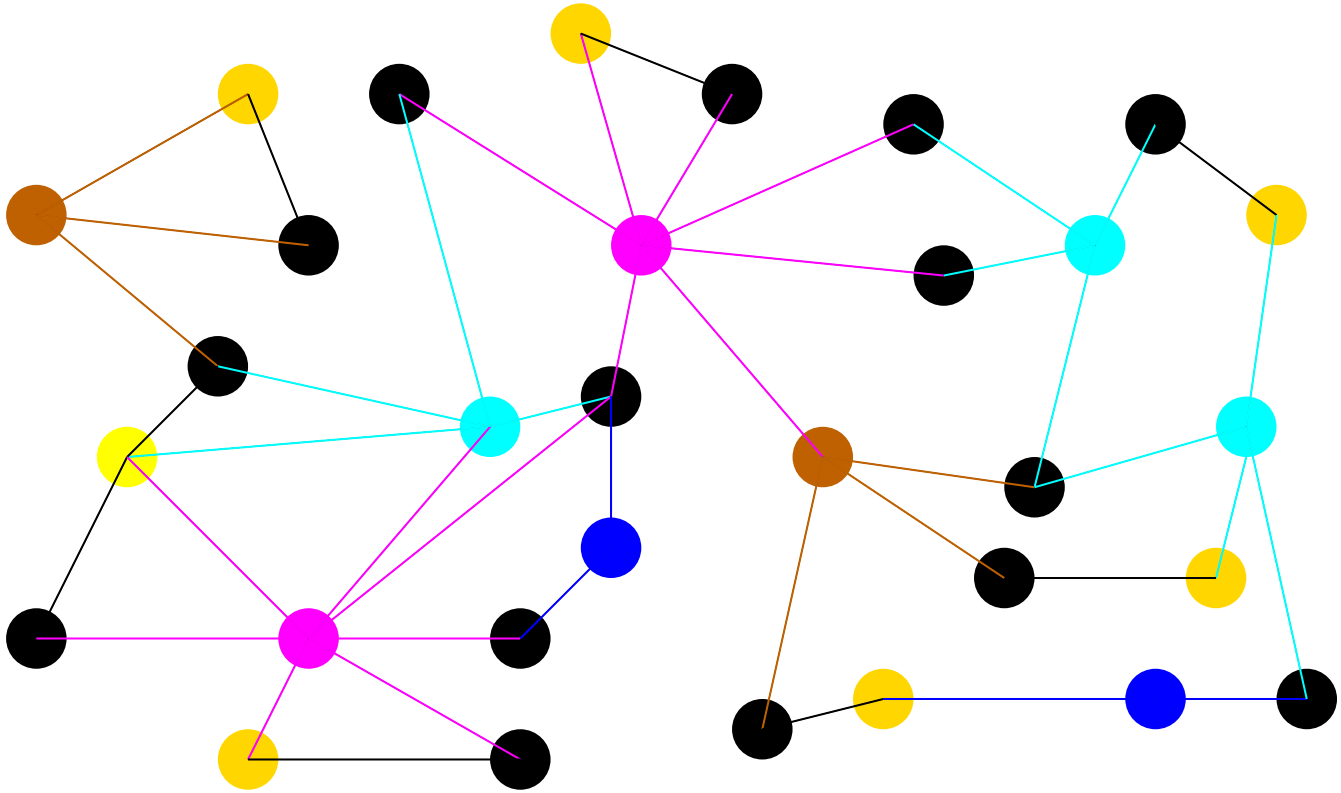
$C \leftarrow \emptyset$
`WHILE` $\exists v \in V : \ \delta(v) > 0$ `DO`
`Pick` v `of largest degree;` $C \leftarrow C \cup \{v\}$`;` $G \leftarrow G - v$
`OD`
`return` C

Wie gut ist GreedyVC?

Ein Beispiel: Wie groß ist ein kleinstmögliches VC ?



Unser Beispiel: Greedy at work. . . Farben kodieren Grad $\leadsto$ 16 VC-Knoten



Ein böses Beispiel (siehe Tafel)

Betrachte folgenden zweifärbbaren Graphen:
Es gibt $r \geq 2$ schwarze Knoten im Graphen.
Die weißen Knoten befinden sich auf r Ebenen:
mit $\left\lfloor \frac{r}{j} \right\rfloor$ Knoten auf Ebene j, $j = 1, \ldots, r$.
Die Kanten definieren r ("möglichst gleichmäßige") Abbildungen der schwarzen Knoten in die weißen Knoten von Ebene j.
GreedyVC wählt z.B. als Erstes den weißen Knoten vom Grad r auf Ebene r, dann die Knoten auf Ebene $r-1$ usf.,
denn ein weißer Knoten auf Ebene j hat mind. Grad j und
nach Löschen der Ebenen $j+1, \ldots, r$ haben die schwarzen Knoten Grad j.
So werden alle $r \cdot \sum_{j=1}^{r}(1/j) \in \Theta(r \log(r))^*$ viele weiße Knoten ausgewählt.
Besser ist jedoch die Auswahl aller r schwarzen Knoten.
Also ist GreedyVC keine c-Approximation für konstantes c.

*Dies ist ein bekanntes Ergebnis über die *harmonische Reihe*.

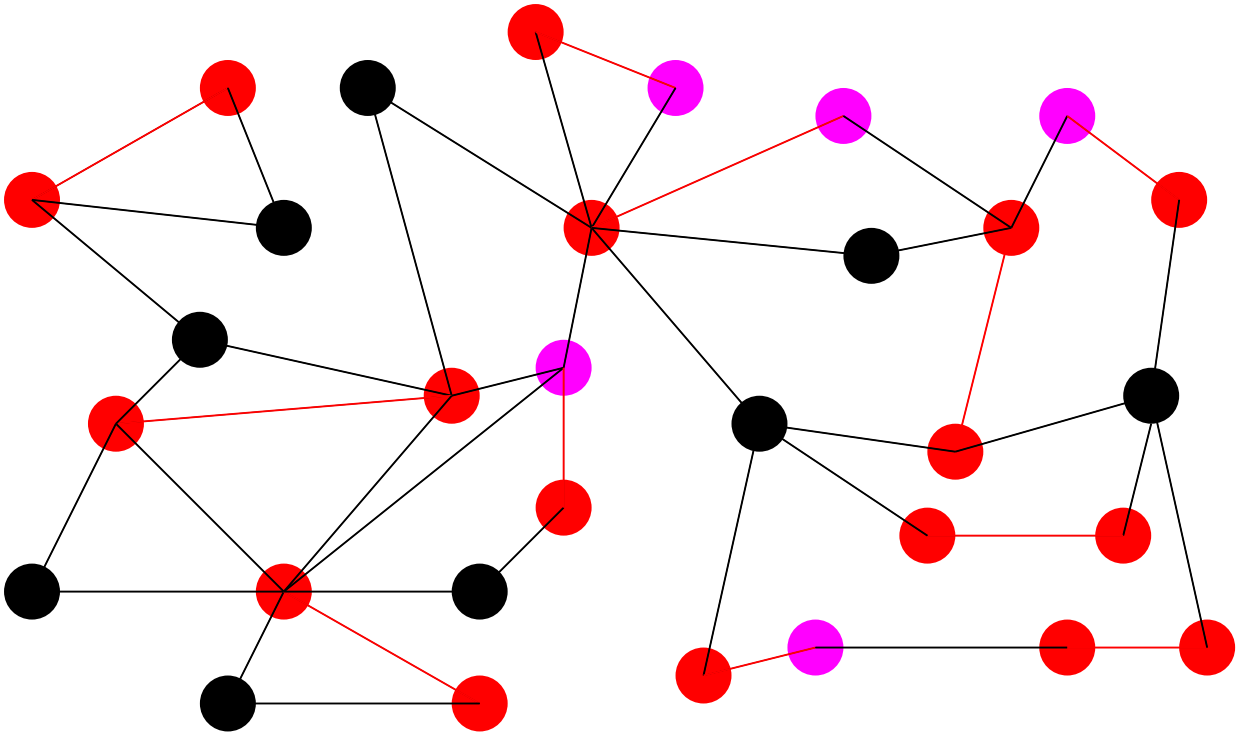
Ein Näherungsverfahren $\mathrm{N1VC}(G = (V, E), C)$

- Falls E leer, gib C aus; exit.

- Nimm irgendeine Kante $e = \{v_1, v_2\}$ aus G
  und berechne $\mathrm{N1VC}(G - \{v_1, v_2\}, C \cup \{v_1, v_2\})$

Lemma: Dies ist ein **2**-Approximations-Verfahren:
Nach Def. muss mind. einer der beiden für Kante e aufgenommenen Knoten in bel. (opt.) Überdeckung C^* liegen.

Unser Beispiel:
magentafarbene Knoten gehören nicht zu *(inklusions-)minimaler* Überdeckung;
die durch Nachverbesserung gefundene Lösung ist sogar fast bestmöglich
Wichtig: Minimal (Greedy) versus Minimum VC (*NP*-hart).

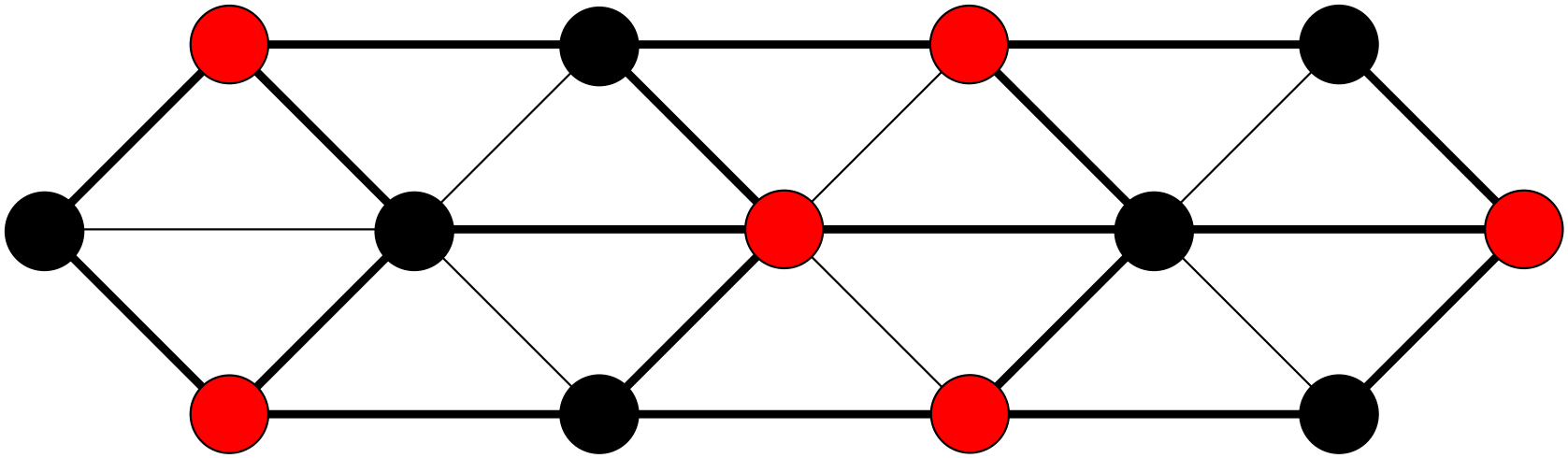


MaxCut

Gegeben: ungerichteter Graph $G = (V, E)$
Gesucht: Größtmöglicher Kantenschnitt, d.h. eigentlich: Partition $(S, V \setminus S)$ von V, sodass die Zahl der Kanten zwischen S und $V \setminus S$ maximal ist.
Natürlich genügt die Angabe von S für einen Schnitt.
Sei dazu $w(S) := |\{e \in E \mid S \cap e \neq \emptyset \wedge e \setminus S \neq \emptyset\}|$.
Für alle Knotenmengen S soll also $w(S)$ maximiert werden.

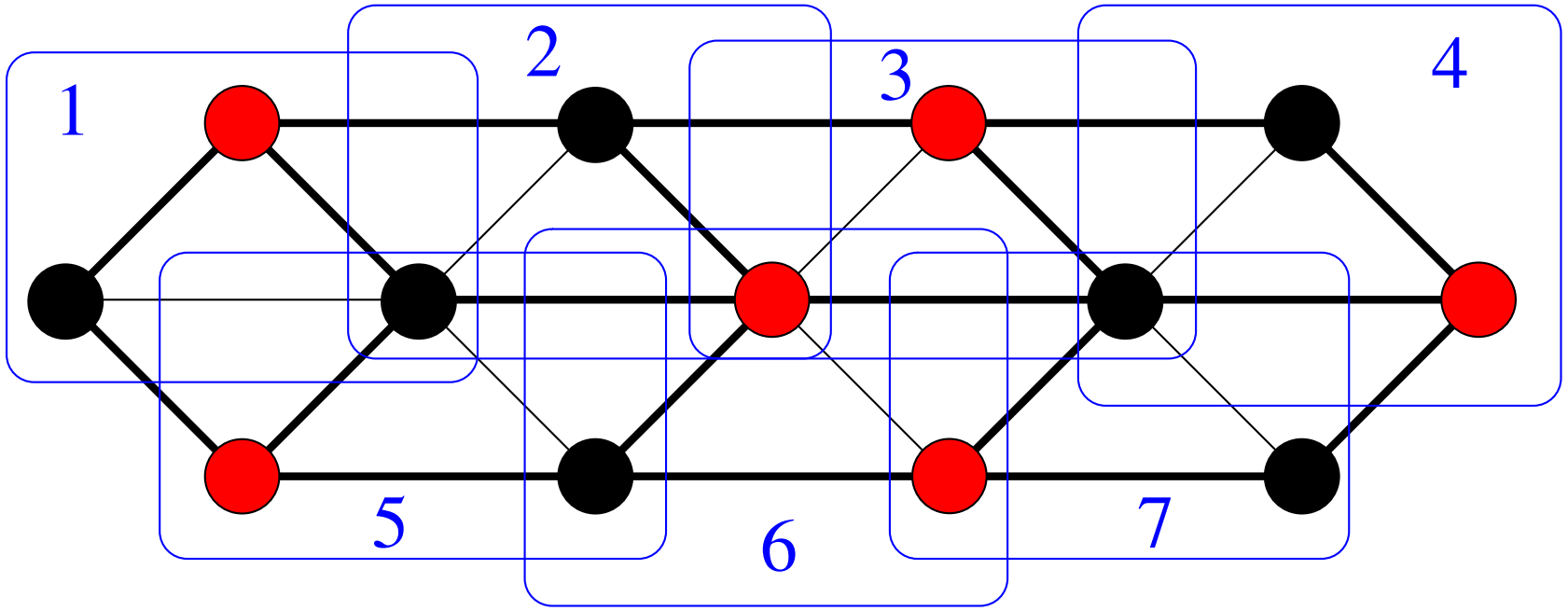
Dazu gibt es ein natürliches Entscheidungsproblem, $k-$Cut.
Mitteilung: $k-$Cut ist NP-vollständig.

Beispiel mit Lösung zu $k-$CUT mit $k = 19$



Ist die angegebene Lösung (Rot-Schwarz-Aufteilung) maximal?

Beispiel mit Lösung zu k−CUT mit $k = 19$



Graph enthält 7 kantendisjunkte Dreiecke

$\rightsquigarrow$ höchstens zwei Kanten jedes dieser Dreiecke kommt in den Schnitt

$\rightsquigarrow$ $k = 19$ ist maximal, da 26 Kanten insgesamt

Eine lokale Suche nach einem Schnitt: LS-MaxCut

$S \leftarrow \emptyset$

`WHILE` $\exists v \in V: \; w(S \,\Delta\, \{v\}) > w(S)$ `DO`

$\quad S \leftarrow S \,\Delta\, \{v\}$

`OD`

`return` S

Beobachte: $\forall v \in V : 2 \cdot |\{e \in E \mid e \subseteq N[v] \wedge |e \cap S| = 1\}| \geq |\{e \in E \mid e \subseteq N[v]\}|$.
Andernfalls wäre nämlich $w(S \,\Delta\, \{v\}) > w(S)$.
Also: Mindestens die Hälfte aller Kanten sind Schnittkanten bzgl. S.
Da trivialerweise $w(S^*) \leq |E|$ für optimalen Schnitt S^*,
ist LS-MaxCut eine 2-Approximation.

Zurück zu den Kreisen

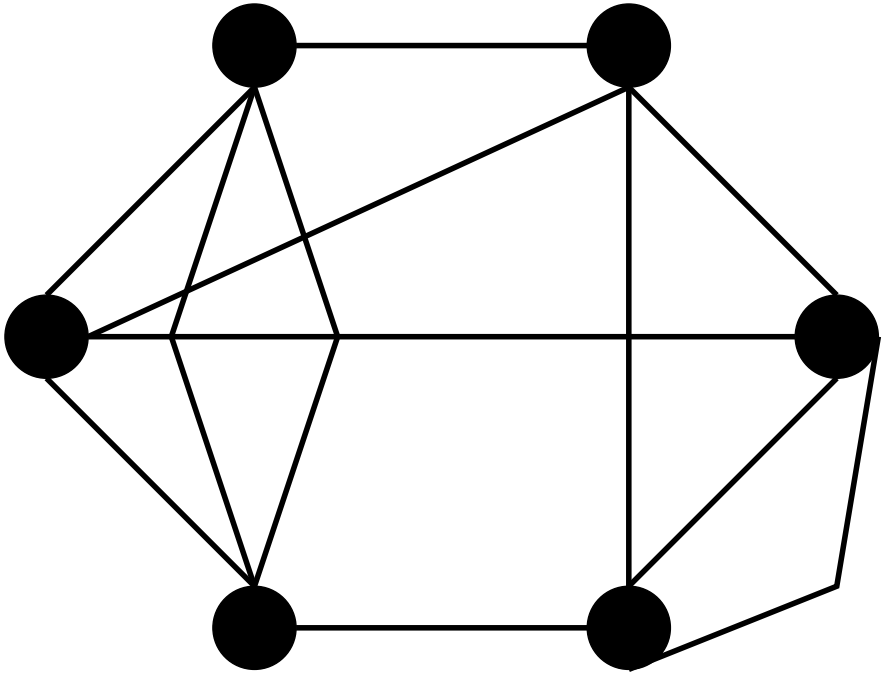
Erinnerung: Ein *Hamiltonkreis* ist eine Folge jeweils benachbarter Knoten, die außer dem Anfangs=End-Knoten keine Dopplungen enthält und die alle Knoten des Graphen beinhaltet.
Die Frage nach der Existenz eines Hamiltonkreises ist NP-hart.

Ein *Eulerkreis* ist eine Folge jeweils benachbarter Kanten, die außer der Anfangs=End-Kante keine Dopplungen enthält und die alle Kanten des Graphen beinhaltet.
Mitteilung: Ein zusammenhängender Graph besitzt einen Eulerkreis genau dann, wenn alle seine Knotengrade gerade sind.
Diese Eigenschaft lässt sich also sehr einfach in Polynomzeit testen.
Dies gilt auch für *Multigraphen*, bei denen Mehrfachkanten zugelassen sind.
Im Folgenden: auch Eulerkreise (auch) als Knotenfolgen...

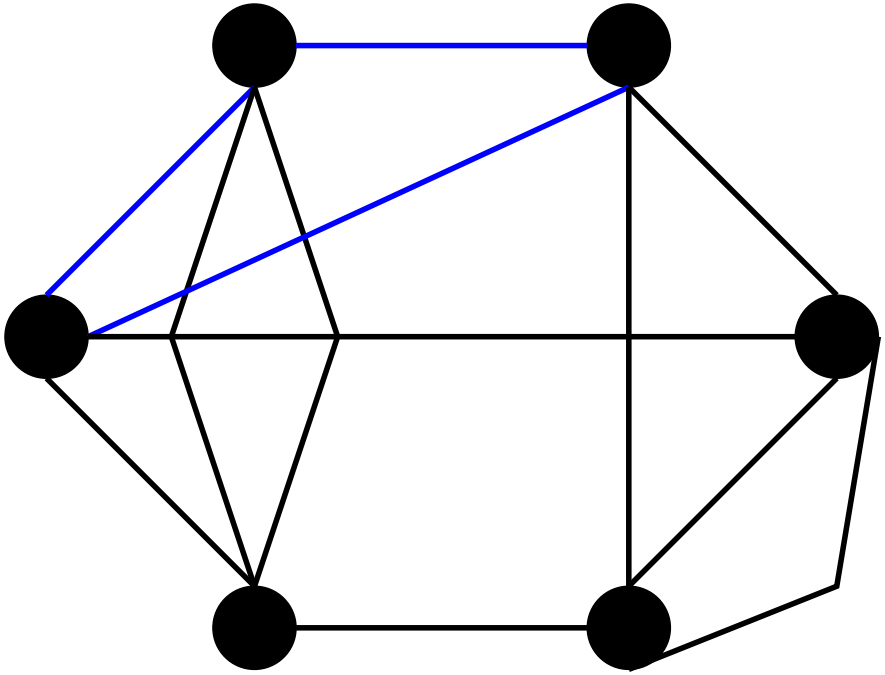
```
EulerKreis(G = (V, E)):
 K ← ∅;
 IF in G  kein Knoten ungeraden Grades existiert
 THEN
   Sei u Knoten kleinsten Grades in G. Setze v ← u.
   REPEAT
     Sei w Knoten kleinsten Grades in der Nachbarschaft von v.
     Füge Kante {v, w} zum Kreis K hinzu und lösche sie aus E.
     Setze v ← w.
   UNTIL v = u
   Lösche isolierte Knoten aus G.
   return Konkatenation aus Folge K und EulerKreis(G).
```

Für Graphen mit nicht-leerer Kantenmenge liefert EulerKreis entweder die leere Menge (Fehlerfall) oder einen gültigen Eulerkreis. Die Konkatenation von Kreisen K und K' wählt zunächst irgendeinen gemeinsamen Knoten und fügt dann K in K' dort ein.

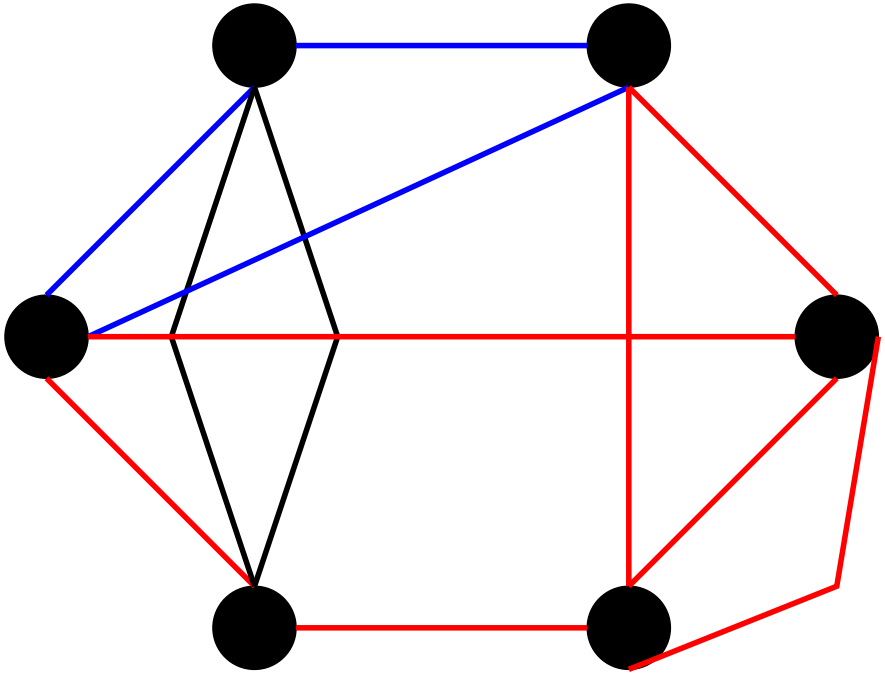
Ein Beispiel: Auffinden von Eulerkreisen



Ein Beispiel: Auffinden von Eulerkreisen

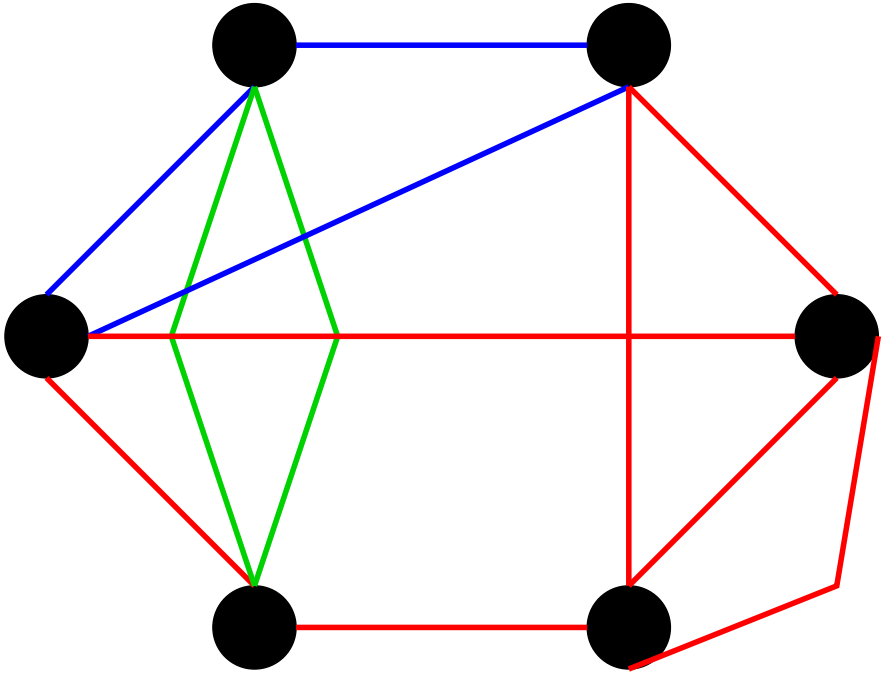


Ein Beispiel: Auffinden von Eulerkreisen



Ein Beispiel: Auffinden von Eulerkreisen

Der grüne Kreis wird in den blau-roten “eingeschoben”.



Zur Korrektheit des Eulerkreisalgorithmus

- Jede Rekursion löscht einen kantendisjunkten Kreis aus dem Graphen.
- $\rightsquigarrow$ Jeder Rekursionsaufruf erniedrigt den Grad jedes Knoten um eine gerade Zahl.
- Jeder Rekursionsaufruf produziert einen *eulerartigen Kreis*, d.h., eine Folge jeweils benachbarter Kanten, die außer der Anfangs=End-Kante keine Dopplungen enthält.
- Dann und nur dann, wenn der Algorithmus ursprünglich mit einem unzusammenhängenden Graphen aufgerufen wird, "versagt" irgendwann die Konkatenationsoperation auf eulerartigen Kreisen.
- Insbesondere: Ist Ursprungs-Graph zusammenhängend, garantiert die Wahl des kleinstgradigen Knotens den Zusammenhang der eulerartigen Kreise und somit die Wohldefiniertheit der Konkatenation: der bisher gewonnene eulerartige Kreis K wird am "Startpunkt" $v \in V(\mathrm{K})$ des "neuen" eulerartigen Kreises K' "aufgeschnitten" und K' wird dort eingefügt.
- Funktioniert die Konkatenation, so ergibt sich wieder ein eulerartiger Kreis.
- Die Rekursion bricht ab, wenn alle Kanten in einem eulerartigen Kreis enthalten sind, wenn also ein Eulerkreis konstruiert wurde.
- Die blaue Wahl des Fortsetzungsknotens ist willkürlich.

Zusammenfassung Eulerkreise

Satz: In einem Multigraphen lässt sich in Polynomzeit ein Eulerkreis bestimmen (so es einen gibt).

Der angegebene Algorithmus leistet das Gewünschte.

Der Grad des Polynoms zur Laufzeitabschätzung hängt von verschiedenen Einzelheiten der Implementierung ab (z.B.: Wahl geeigneter Datenstrukturen) und wird daher hier nicht weiter erörtert.

Spannbäume

$G' = (V', E')$ heißt *Teilgraph* von $G = (V, E)$, wenn G' Graph mit $V' \subseteq V$ und $E' \subseteq E$; G' heißt *(auf-)spannender Teilgraph*, falls $V' = V$.

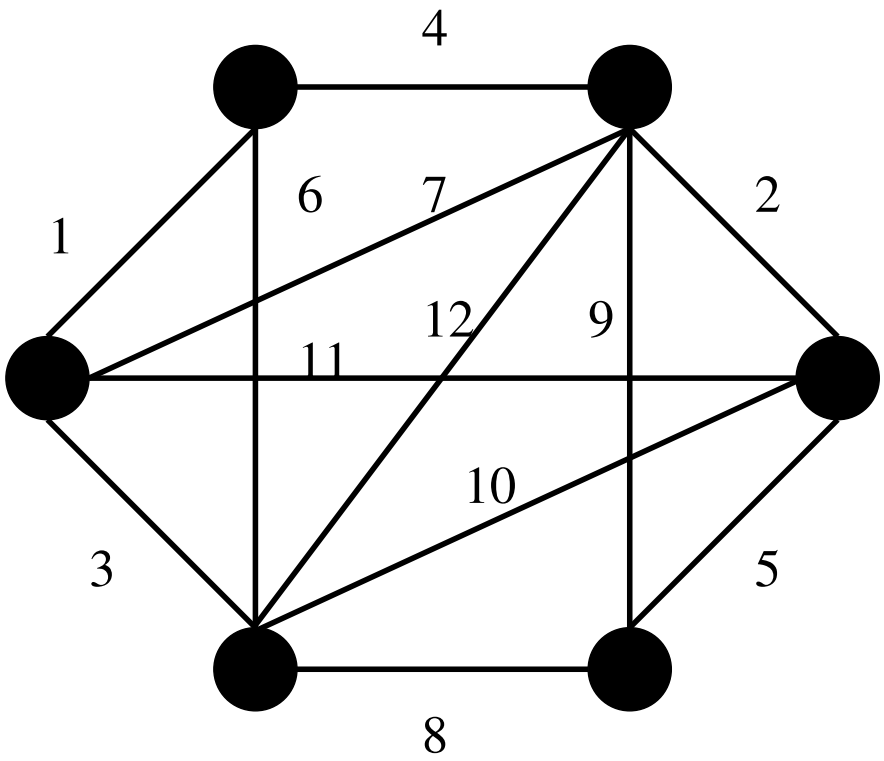
Ein kreisfreier zusammenhängender aufspannender Teilgraph $G' = (V', E')$ von $G = (V, E)$ heißt auch *Spannbaum* oder *Gerüst*.
Es ist oft bequem, Spannbäume als Kantenmengen T zu sehen.

Ist $G = (V, E)$ ein *kantengewichteter Graph*, d.h., es gibt zusätzlich Gewichtsfunktion $w : E \to \mathbb{N}$, so heißt ein Spannbaum $T \subseteq E$ von G *minimal* gdw. $w(T) \leq w(T')$ für bel. Spannbaum $T' \subseteq E$ gilt.
Dabei ist $w(T) = \sum_{e \in T} w(e)$.

Satz: Ein minimaler Spannbaum kann in Polynomzeit bestimmt werden.
Algorithmus von Kruskal (Skizze): (Korrektheit später!)

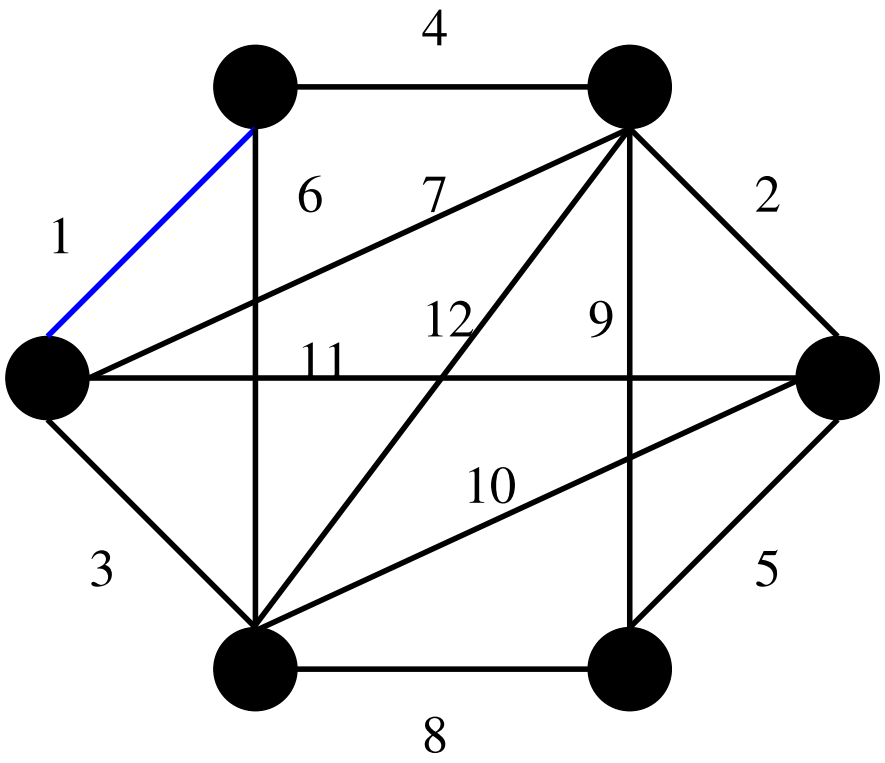
Sortiere E aufsteigend: $E = \{e_1 \leq e_2 \leq \cdots \leq e_m\}$.
Setze $T \leftarrow \emptyset$.
FOR $i \leftarrow 1$ TO m DO
  Sei $e_i = \{u, v\}$.
  IF NOT u und v sind durch Kantenzug aus T verbunden
  THEN $T \leftarrow T \cup \{e_i\}$
  FI
OD
return T

Ein Beispiel: Auffinden von minimalen Spannbäumen



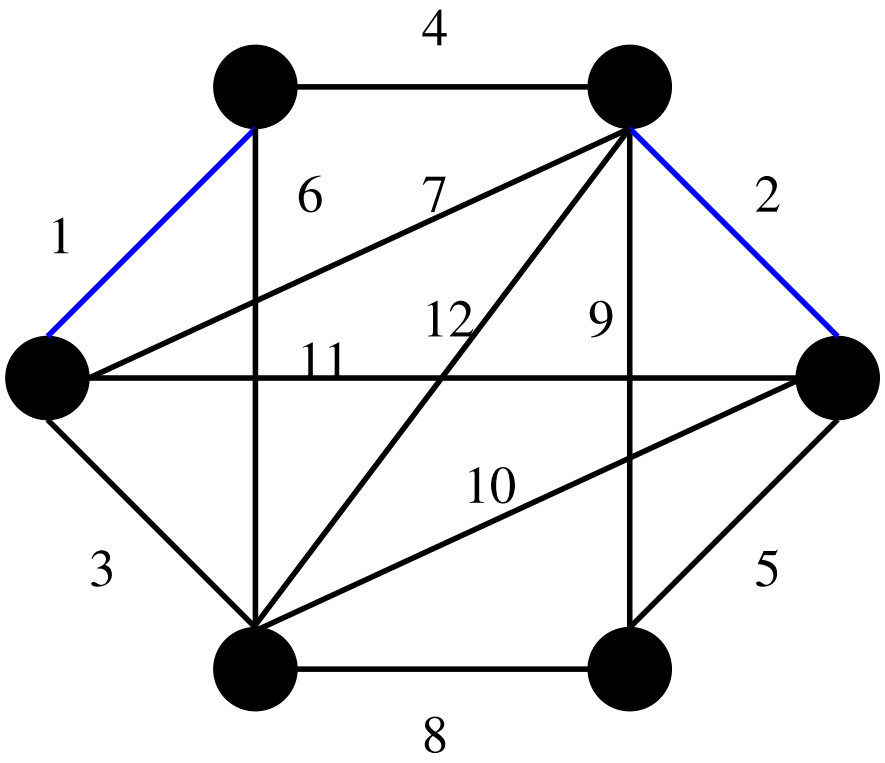
Ein Beispiel: Auffinden von minimalen Spannbäumen

Einfügen der billigsten Kante



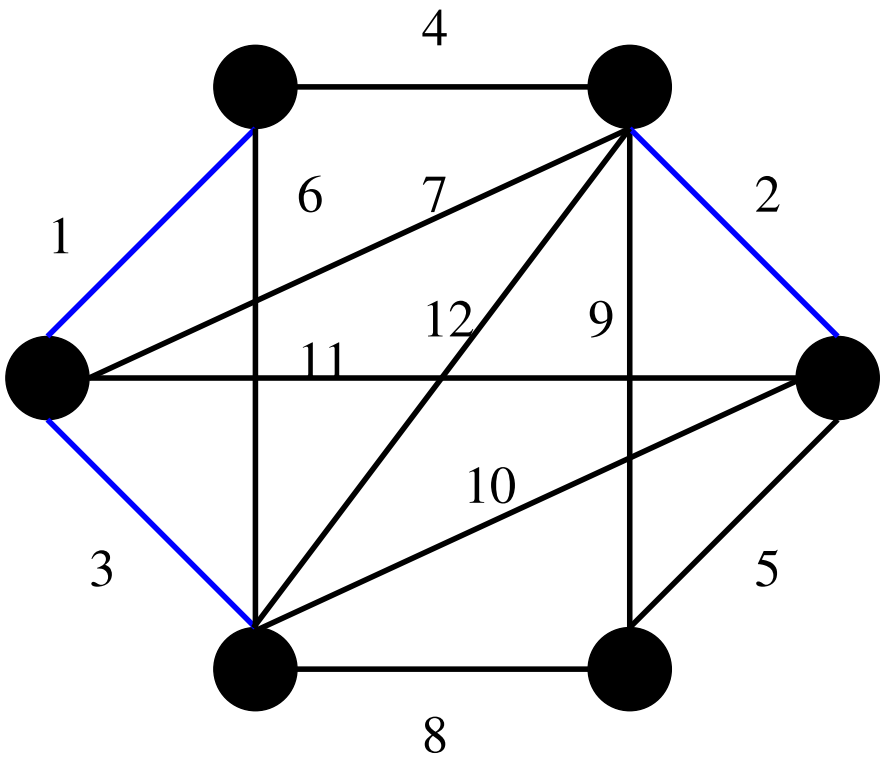
Ein Beispiel: Auffinden von minimalen Spannbäumen

Einfügen der zweitbilligsten Kante



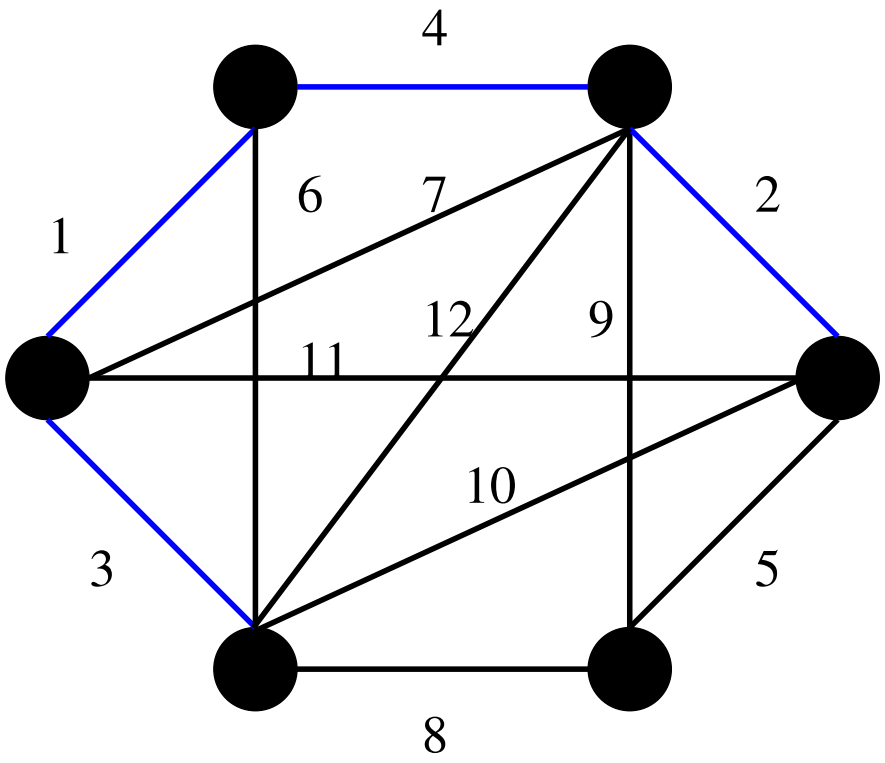
Ein Beispiel: Auffinden von minimalen Spannbäumen

Einfügen der drittbilligsten Kante



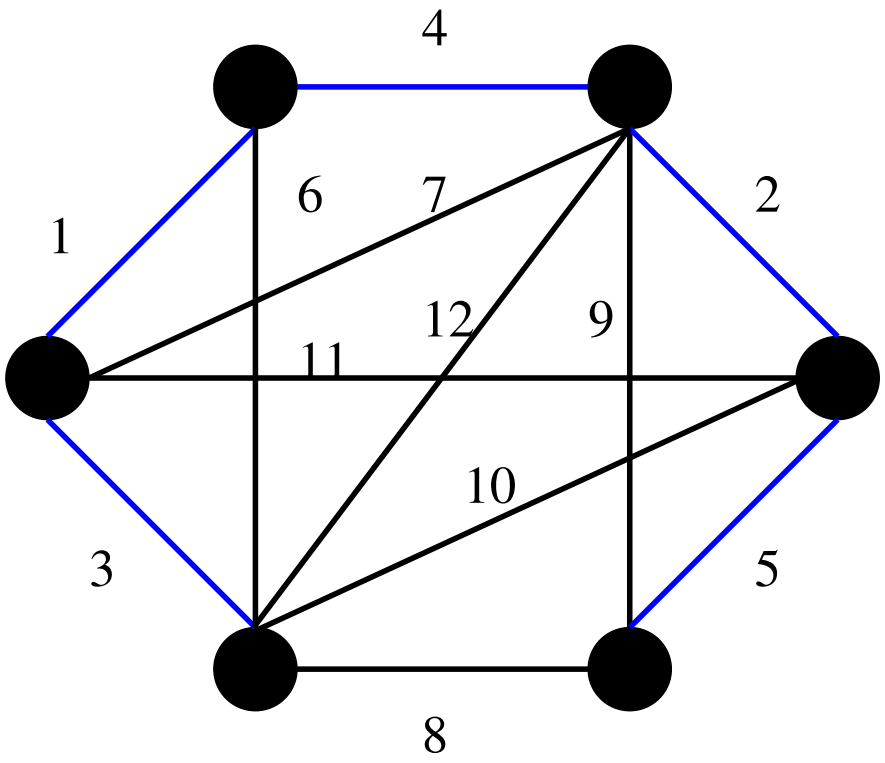
Ein Beispiel: Auffinden von minimalen Spannbäumen

Einfügen der viertbilligsten Kante



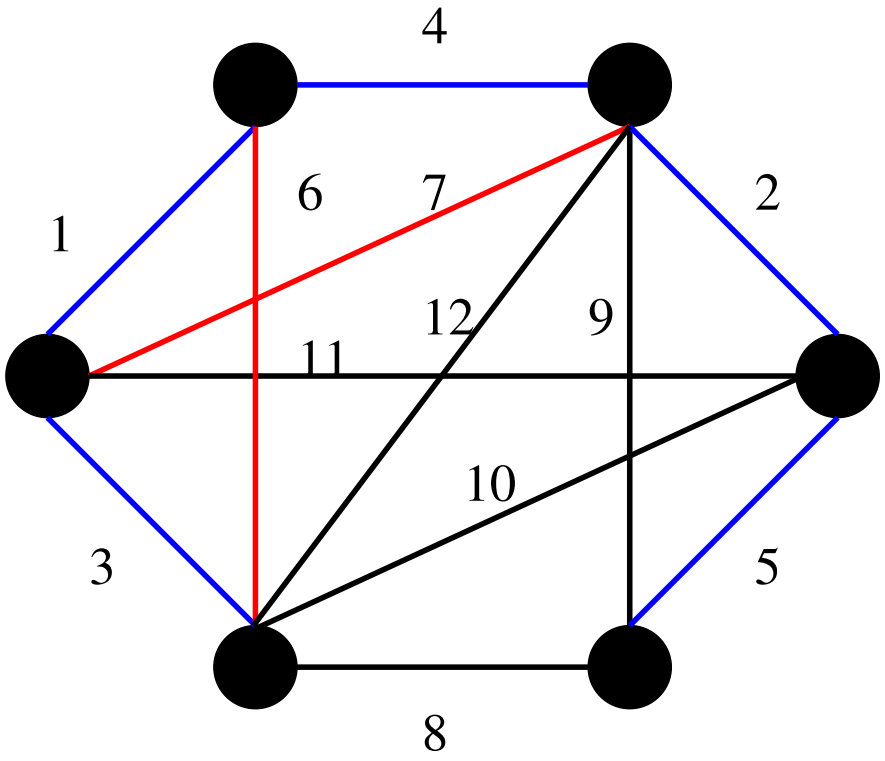
Ein Beispiel: Auffinden von minimalen Spannbäumen

Einfügen der fünftbilligsten Kante



Ein Beispiel: Auffinden von minimalen Spannbäumen

Einfügen weiterer Kanten scheitern: Es entstehen Kreise



Eine Approximation für metrisches TSP nach Christofides

Gegeben: kantengewichteter Graph $G = (V, E)$;
die Kantengewichte genügen der Dreiecksungleichung.

```
Berechne minimalen Spannbaum T.
Konstruiere hieraus Multigraphen M
  durch Verdoppeln aller Kanten (und deren Gewichten).
Bestimme Eulerkreis K in M.
Konstruiere hieraus Tour C wie folgt:
  Laufe Eulerkreis Punkt für Punkt ab:
  Wurde ein Knoten bereits besucht
     (und der wurde Kreis nicht geschlossen),
       so überspringe diesen Punkt und nimm Abkürzung.
```

Eine Approximation für metrisches TSP nach Christofides

Satz: Der angebene Algorithmus ist eine 2-Approximation.

Beweis: Klar (?): Der Algorithmus liefert stets eine zulässige Tour in Polynomzeit.
Sei C^* eine optimale Tour. Durch Fortlassen einer Kante entsteht hieraus ein Spannbaum.
Daher gilt: $w(T) \leq w(C^*)$, wobei T der berechnete MST.
Trivialerweise gilt für den Eulerkreis K nach Konstruktion: $w(K) = 2 \cdot w(T)$.
Da die Tour C durch "Abkürzungen" aus K entsteht,
gilt wegen der Gültigkeit der Dreiecksungleichung:

$$w(C) \leq w(K) \leq 2 \cdot w(C^*).$$

Naheliegende Verbesserung: Schlaueres Sicherstellen der Existenz von Eulerkreisen: Füge nur Kanten ein bei ungerade-gradigen Knoten.