Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement Informatik
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger

8. Dezember 2016

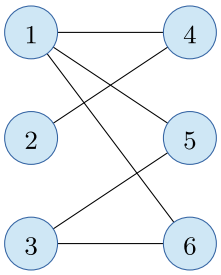
Algorithmen & Datenstrukturen Lösungen zu Blatt 11 HS 16

Lösung 11.1 *Eigenschaften von Beispielen für Graphen.*

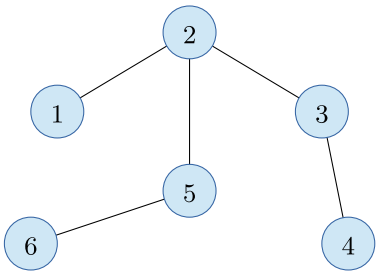
a) (i)
$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

(ii)
$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

b) (iii)



(iv)



c) (i) Gerichtet, nicht bipartit, nicht kreisfrei, Schleife um den Knoten 2,

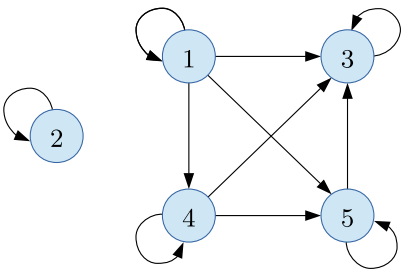
(ii) Ungerichtet, nicht bipartit, nicht kreisfrei, keine Schleifen,

(iii) Ungerichtet, bipartit, nicht kreisfrei, keine Schleifen,

(iv) Ungerichtet, bipartit, kreisfrei, keine Schleifen.

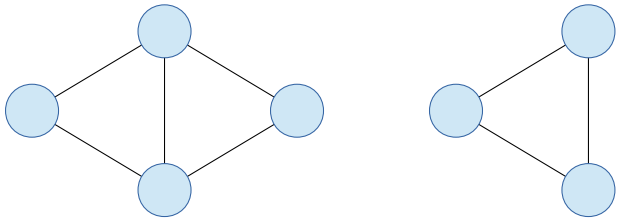
d) Nur die Graphen (ii), (iii) und (iv) sind ungerichtet. Sie alle sind zusammenhängend, da zwischen jedem Paar zweier Knoten v und w ein Weg von v nach w existiert. Der Graph (ii) besitzt einen Eulerkreis, zum Beispiel $\langle 1, 4, 3, 2, 5, 3, 1\rangle$. Damit besitzt er natürlich auch einen Eulerschen Weg (bei dem Anfangs- und Endknoten übereinstimmen). Der Graph (iii) besitzt keinen Eulerkreis, da es Knoten mit ungeradem Grad gibt. Da aber nur zwei Knoten einen ungeraden Grad haben, gibt es einen Eulerschen Weg, z.B. $\langle 1, 5, 3, 6, 1, 4, 2\rangle$. Der Graph (iv) besitzt weder einen Eulerkreis, noch einen Eulerschen Weg, da es vier Knoten mit ungeradem Grad gibt.

e)



$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

f) Hier gibt es verschiedene Lösungen, zum Beispiel



Es existieren noch weitere Lösungen, z.B. solche, bei denen einige Knoten Schleifen haben.

Einen ungerichteten Graphen mit 7 Knoten, von denen genau einer Grad 3 und alle anderen Grad 2 haben, gibt es nicht, da die Summe aller Knotengrade immer gerade sein muss.

Lösung 11.2 *Eigenschaften von Graphen.*

a) Jeder ungerichtete Graph ohne Schleifen hat höchstens $\binom{n}{2}$ viele Kanten, denn es gibt genau so viele zweielementige Teilmengen (Kanten) einer n-elementen Grundmenge (Knoten). Sind auch Schleifen erlaubt, kommen noch n Kanten hinzu, d.h. ein ungerichteter Graph mit Schleifen hat höchstens

$$\binom{n}{2} + n = \frac{n(n-1)}{2} + \frac{2n}{2} = \frac{n(n+1)}{2} = \binom{n+1}{2} \tag{1}$$

viele Kanten.

Ein gerichteter Graph hat, wenn er Schleifen hat, maximal n^2 viele Kanten (jeder Knoten kann eine Kante zu jedem anderen Knoten haben). Ist ein gerichteter Graph schleifenlos, dann jeder Knoten mit maximal $n-1$ anderen Knoten verbunden sein und kann daher maximal $n(n-1)$ viele Kanten haben.

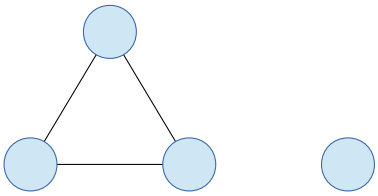
b) *Induktionsverankerung* ($n = 1$): Jeder Graph mit genau einem Knoten ist zusammenhängend. Kreisfrei ist er genau dann, wenn er keine Schleifen und damit auch keine Kanten hat. Jeder Baum mit $n = 1$ Knoten hat also $n - 1 = 0$ Kanten.

Induktionshypothese: Jeder Baum mit n Knoten habe genau $n-1$ Kanten.

Induktionsschritt ($n \to n+1$): Betrachte einen beliebigen Baum $T = (V, E)$ mit $n+1$ Knoten. Wir zeigen zunächst, dass jeder Baum mindestens einen Knoten mit Grad 1 enthält. Angenommen, es gäbe einen Baum, in dem jeder Knoten Grad 2 oder grösser hätte. Dann könnten wir einen beliebigen Knoten auswählen, von dort aus über eine bisher unbenutzte Kante zu einem Nachbarn wechseln, von dort erneut über eine bisher unbenutzte Kante zu einem Nachbarn wechseln, usw. Nach spätestens n Schritten würden wir aber zu einem Knoten gelangen, den wir bereits früher besucht haben. Also wäre der Graph nicht kreisfrei und damit auch kein Baum.

Sei nun v ein beliebiger Knoten in T mit Grad 1, und sei $e = \{v, w\}$ die entsprechende Kante, die v enthält. Da v Grad 1 hat, ist die Kante e eindeutig bestimmt. Entfernen wir v und e aus T, dann erhalten wir den Graph $T' = (V', E')$ mit $V' = V \backslash \{v\}$ und $E' = E \backslash \{\{v, w\}\}$. Dieser muss sowohl zusammenhängend als auch kreisfrei sein (sonst wäre T nicht zusammenhängend oder nicht kreisfrei), also ist T' ein Baum. Da dieser genau einen Knoten weniger als T hat, hat er gemäss Induktionshypothese genau $n-1$ Kanten. Da T genau eine Kante mehr als T' hat (nämlich $\{v, w\}$), hat T damit $n-1+1 = (n+1)-1$ viele Kanten. ∎

c) Die Aussage ist falsch, wie etwa der untenstehende, nicht kreisfreie Graph mit $n = 4$ Knoten und $n - 1 = 3$ Kanten zeigt:



Lösung 11.3 *Zyklenfreiheit testen.*

a) Ein Zyklus der Länge 1 ist eine Schleife. Ein Graph hat genau dann eine Schleife um den Knoten v_i, wenn die Adjazenzmatrix im Diagonaleintrag a_{ii} eine 1 hat. Um zu prüfen, ob der Graph eine Schleife hat, genügt es also, alle Diagonaleinträge a_{ii} für $i = 1, \ldots, n$ anzuschauen, und zu prüfen, ob dort eine 1 steht.

b) Hier reicht es, die transitive Hülle des Graphen zu berechnen und am Ende zu prüfen, ob es einen Diagonaleintrag mit Wert 1 gibt. Anders als in der Vorlesung soll die Hülle also nur transitiv und nicht notwendigerweise reflexiv sein. Zur Berechnung können wir den in der Vorlesung vorgestellten Algorithmus zur Berechnung der reflexiven und transitiven Hülle leicht modifizieren: Zunächst entfernen wir die Anweisung $a_{k,k} \leftarrow 1$ in Schritt 2, und prüfen dann später ob es einen Diagonaleintrag mit Wert 1 gibt.

HAS-CYCLE($A_G = (a_{ij})_{1 \le i,j \le n}$)

```
1 for k ← 1, ..., n do
2     for i ← 1, ..., n do
3         for j ← 1, ..., n do
4             if a_ik = 1 and a_kj = 1 then a_ij ← 1
5 for k ← 1, ..., n do
6     if a_kk = 1 then return "Zyklus gefunden"
7 return "zyklenfrei"
```

Man sieht sofort, dass die Laufzeit des Verfahrens noch immer in $\Theta(n^3)$ liegt.

c) Für diese Aufgabe brauchen wir sowohl die ursprünglich gegebene Adjazenzmatrix A_G als auch die vom Algorithmus in b) berechnete transitive Hülle (im Folgenden B_G genannt), bei der ein Eintrag b_{ij} genau dann 1 ist, wenn es einen gerichteten Weg von i nach j gibt.

Wir beobachten nun folgendes: Wenn ein Diagonaleintrag b_{kk} in B_G den Wert 1 hat, dann gibt es einen Kreis (und damit insbesondere einen Zyklus), der v_k enthält. Also hat v_k mindestens einen Nachbarn v_j, von dem aus v_k wieder erreichbar ist. Es gibt also ein j mit $a_{kj} = 1$ und $b_{jk} = 1$. Der Knoten v_j hat nun erneut einen Nachbarn, von dem aus v_k erreicht werden kann. Um irgendeinen Kreis zu finden, reicht es also prinzipiell aus, bei v_k zu starten und stets *irgendeinem* Nachbarn zu folgen, von dem aus v_k erreicht werden kann. Ein solcher existiert immer, wenn da der aktuell betrachtete Knoten immer auf einem Kreis liegt, der v_k enthält.

Es kann aber passieren, dass wir auf einen Knoten stossen, den wir bereits früher gefunden haben. Dann aber haben wir einen anderen Kreis gefunden, der wir entsprechend ausgeben können. Dazu müssen wir nur speichern, in welcher Reihenfolge wir die bisherigen Knoten

angeschaut haben, d.h., wir speichern den bisherigen Weg W. Aus diesem kann dann ein Kreis extrahiert werden. Für einen Weg W und einen Knoten v_i bezeichne im Folgenden $W \oplus \langle v_i \rangle$ den Weg W, an dessen Ende v_i angehängt wird.

EXTRACT-CYCLE($A_G = (a_{ij})_{1 \le i,j \le n}, B_G = (a_{ij})_{1 \le i,j \le n}$)

```
1  for j ← 1, ..., n do BESUCHT[j] ← 0
2  i ← k                                  ▷ Aktuell betrachteter Knoten
3  W ← ⟨v_k⟩                              ▷ Bisher betrachteter Weg
4  repeat
5      BESUCHT[i] ← 1                     ▷ Markiere aktuellen Knoten als besucht
6      j ← 1                              ▷ Suche einen Nachbarn v_j von v_i,
7      while a_ij = 0 or b_jk = 0 do      ▷ von dem aus v_k erreicht werden kann
8          j ← j + 1
9      i ← j                              ▷ Fahre mit v_j fort
10     W ← W ⊕ ⟨v_i⟩                       ▷ Füge aktuellen Knoten dem Weg hinzu
11 until BESUCHT[i] = 1
12 Entferne jeweils den ersten Knoten aus W, bis dieser exakt v_i ist.
13 return W
```

Da wir in jedem Schritt einen neuen Knoten als besucht markieren, wird die Schleife in den Schritten 4-11 höchstens n Mal durchlaufen. Die Schritte 1, 7 und 8 (zusammen) sowie 12 brauchen jeweils insgesamt Zeit $\mathcal{O}(n)$, alle anderen Schritte können in konstanter Zeit ausgeführt werden. Damit liegt die Gesamtlaufzeit dieses Algorithmus in $\mathcal{O}(n^2)$.