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

15. Dezember 2016

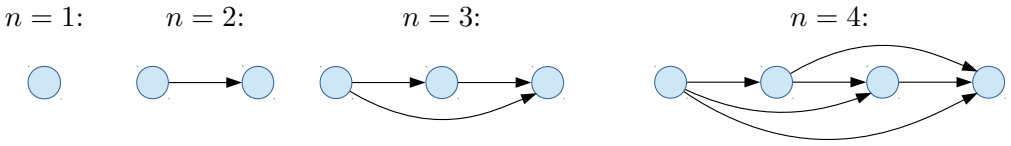
Datenstrukturen & Algorithmen Lösungen zu Blatt 12 HS 16

Lösung 12.1 *Topologisches Sortieren und Zusammenhangskomponenten.*

a) Da G die Kreise (D, B, C, D), (D, B, F, D) und (D, B, C, F, D) enthält, muss mindestens eine Kante entfernt werden, damit G' kreisfrei ist. Entfernen wir nun die Kante (D, B) (d.h., wählen wir $E' = \{(D, B)\}$), dann ist der Graph $G' = (V, E \backslash E')$ kreisfrei und kann daher topologisch sortiert werden. Offenbar ist E' auch kleinstmöglich, und in diesem Fall sogar die einzige Wahl (enthielte E' eine andere Kante als (D, B), dann wäre (D, B) noch immer Teil von mindestens einem Kreis und G' damit nicht kreisfrei).

b) Eine topologische Sortierung für G' ist A, B, C, F, D. Für den Graphen G' ist dies sogar die einzig mögliche topologische Sortierung.

c) Damit ein Graph topologisch sortiert werden kann, muss er kreisfrei sein. Die kreisfreien Graphen mit maximaler Kantenanzahl für $n = 1, 2, 3, 4$ Knoten sind:



Wir erhalten die Vermutung, dass im Allgemeinen jeder kreisfreie Graph nicht mehr als $\sum_{i=1}^{n-1} i = n(n-1)/2$ viele Kanten enthält, was wir mit vollständiger Induktion über n beweisen.

Induktionsverankerung ($n = 1$): Jeder kreisfreie Graph mit einem Knoten hat genau (und damit auch höchstens) $0 = n(n-1)/2$ viele Kanten.

Induktionshypothese: Jeder kreisfreie Graph mit n Knoten habe höchstens $n(n-1)/2$ viele Kanten.

Induktionsschritt ($n \to n+1$): Betrachte einen beliebigen kreisfreien Graphen $G = (V, E)$ mit $n+1$ Knoten. Da der Graph kreisfrei ist, muss er (wie in der Vorlesung gezeigt) mindestens einen Knoten v mit Eingangsgrad 0 haben. Entfernen wir diesen und alle zu ihm inzidenten Kanten aus G, dann erhalten wir einen neuen Graphen G' mit n Knoten, der ebenfalls kreisfrei ist (durch die Entfernung von Kanten können keine neuen Kreise entstehen). Gemäss der Induktionshypothese hat G' höchstens $n(n-1)/2$ viele Kanten. Der aus G entfernte Knoten v kann mit maximal n anderen Knoten (genau denen in G') verbunden gewesen sein. Damit hat der ursprünglich gegebene Graph G höchstens $n(n-1)/2 + n = n(n-1)/2 + 2n/2 = n(n+1)/2$ viele Kanten. ∎

Tatsächlich ist diese Schranke auch bestmöglich, denn es gibt einen kreisfreien Graphen mit exakt so vielen Kanten. Sei die Knotenmenge $V = \{v_1, \ldots, v_n\}$ gegeben. Jetzt erzeugen wir für jedes $i \in \{1, \ldots, n-1\}$ und jedes $j \in \{i+1, \ldots, n\}$ die Kante (v_i, v_j). Der entstehende

Graph ist kreisfrei (denn von jedem Knoten v_i gibt es nur Kanten zu Knoten v_j mit $j > i$, also gibt es keinen Weg zu v_i zurück), und er hat exakt $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ viele Kanten.

d) Zwei Knoten befinden sich genau dann in der gleichen Zusammenhangskomponente, wenn zwischen ihnen ein Weg existiert. Damit ist eine Zusammenhangskomponente also immer zusammenhängend. Hat eine Zusammenhangskomponente minimale Kantenanzahl, dann kann sie keine Kreise haben, denn das Entfernen einer Kante auf dem Kreis würde den Zusammenhang nicht zunichtemachen. Zusammenhangskomponenten mit minimaler Kantenanzahl sind damit ungerichtete, kreisfreie, zusammenhängende Graphen, also Bäume. Diese haben genau eine Kante weniger als Knoten.

Hat ein Graph k Zusammenhangskomponenten, die jeweils die Knoten $V_1, \ldots, V_k$ enthalten (wobei $V_i \cap V_j = \emptyset$ und $\cup_{i=1}^{n} V_i = V$ sind), dann enthält dieser Graph mindestens

$$\sum_{i=1}^{k}(|V_i| - 1) = \big(\sum_{i=1}^{k} |V_i|\big) - k = |V| - k = n - k \tag{1}$$

viele Kanten.

Lösung 12.2 *Tiefen- und Breitensuche.*

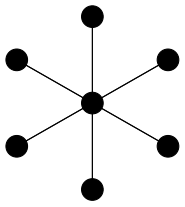
a) Tiefenordnung: A, B, C, D, E, F, H, G

Breitenordnung: A, B, F, C, H, D, G, E

b) Nein, denn in diesem Graphen besucht eine Tiefensuche für jeden Startknoten die Knoten in einer anderen Reihenfolge als eine Breitensuche. Die folgenden Reihenfolgen werden von einer Tiefensuche erzeugt und können von einer Breitensuche nicht erzeugt werden:

- Start bei A: $A, B, C, \ldots$
- Start bei B: $B, C, D, E, \ldots$
- Start bei C: $C, D, E, \ldots$
- Start bei D: $D, E, A, \ldots$
- Start bei E: $E, A, B, \ldots$
- Start bei F: $F, H, C, D, E, \ldots$
- Start bei G: $G, H, C, \ldots$
- Start bei H: $H, C, D, E, \ldots$

c) Ein Beispiel ist ein sternförmiger Graph (siehe unten). Beginnen wir die Traversierung in der Mitte, dann ist klar, dass jede Breitenordnung eine Tiefenordnung ist und umgekehrt. Wenn wir nicht in der Mitte beginnen, dann ist der nächste Knoten in jedem Fall die Mitte, und wir befinden uns wieder in der zuvorgenannten Situation.



d) Sowohl Breiten- als auch Tiefensuche müssen für jeden Knoten einmal alle Nachbarn durchgehen. Mit Adjazenzlisten benötigt das asymptotisch genau so viele Schritte, wie ein Knoten Nachbarn hat. Die Laufzeit ist in diesem Fall in $\mathcal{O}(|V| + |E|)$, also für zusammenhängende Graphen in $\mathcal{O}(|E|)$.

Mit einer Adjazenzmatrix kann man die Nachbarn eines Knotens nur finden, indem man alle Einträge der entsprechenden Zeile betrachtet. Da dies für jeden Knoten getan werden muss, ist die Laufzeit in $\Omega(|V|^2)$.

Bei einem sehr dichten Graphen mit $|E| \in \Theta(|V|^2)$, ist die asymptotische Laufzeit gleich. Bei "dünnen" Graphen (wenn z.B. $|E| \in \mathcal{O}(|V|)$ gilt) führt die Verwendung einer Adjazenzmatrix allerdings zu einer erheblich schlechteren asymptotischen Laufzeit.

Lösung 12.3 *Schwarze Löcher.*

Dieses Problem kann analog zum Problem "Star in einer Menge finden" gelöst werden, welches in der ersten Vorlesung vorgestellt wurde.

Wir beobachten zunächst, dass es maximal ein schwarzes Loch geben kann: Sei v ein schwarzes Loch. Nach Definition ist der Ausgangsgrad von v genau 0. Also ist $(v, v) \notin E$. Da der Eingangsgrad genau $|V| - 1$ beträgt, gibt es für alle $v' \in V \backslash \{v\}$ eine Kante $(v', v) \in E$ und der Ausgangsgrad jedes Knotens $v' \in V \backslash \{v\}$ beträgt mindestens 1. Damit kann v' kein weiteres schwarzes Loch sein.

Der Algorithmus besteht aus zwei Phasen. Zunächst finden wir einen Kandidaten v für ein schwarzes Loch mit $|V| - 1$ Vergleichen. Danach prüfen wir mit $2|V| - 1$ Vergleichen, ob v wirklich ein schwarzes Loch ist. Sei $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ und der zugehörigen Adjazenzmatrix $A_G = (a_{ij})$ mit $a_{ij} \in \{0, 1\}$, $a_{ij} = 1 \Leftrightarrow (v_i, v_j) \in E$, gegeben. Zur Bestimmung eines Kandidaten benutzen wir zwei Variablen i und j, die mit $i \leftarrow 1$ und $j \leftarrow n$ initialisiert werden. Danach wird der Eintrag a_{ij} geprüft. Ist $a_{ij} = 0$, dann ist $(v_i, v_j) \notin E$ und v_j sicher kein schwarzes Loch (denn der Eingangsgrad von v_j ist kleiner als $n-1$, falls $(v_j, v_j) \notin E$). Also setzen wir $j \leftarrow j - 1$. Ist dagegen $a_{ij} = 1$, dann ist $(v_i, v_j) \in E$ und v_i kein schwarzes Loch (denn der Ausgangsgrad von v_i ist mindestens 1). Also setzen wir $i \leftarrow i + 1$. Nach $n - 1$ solchen Vergleichen ist $i = j$ und v_i wird unser Kandidat für ein schwarzes Loch. Nun prüfen wir noch, ob v_i wirklich ein schwarzes Loch ist, indem wir den Eingangsgrad und den Ausgangsgrad von v_i berechnen. Dazu müssen wir lediglich prüfen, ob $a_{ij} = 0$ für $j = 1, \ldots, n$ und $a_{ji} = 1$ für $j = 1, \ldots, n$, $j \neq i$, sind. Ist dies der Fall, dann ist v_i das gesuchte schwarze Loch. Ansonsten besitzt G kein schwarzes Loch.

Die Korrektheit des Verfahrens folgt aus der Beobachtung, dass die Zeiger i bzw. j nur dann aktualisiert werden, wenn v_i bzw. v_j *kein* schwarzes Loch sind. Ist umgekehrt i so gesetzt, dass v_i ein schwarzes Loch ist, dann ist $a_{ij} = 0$ für alle $j = 1, \ldots, n$, also wird i niemals vergrössert (sondern nur noch j verkleinert). Analog wird j niemals verkleinert, wenn v_j ein schwarzes Loch ist, da in diesem Fall $a_{ij} = 1$ ist für alle $i = 1, \ldots, n$, $i \neq j$.