RWTH Aachen **SS 2016**
Lehrgebiet Theoretische Informatik Lösungsvorschlag 8
Rossmanith–Hensel–Kuinke–Sánchez 13. Juni 2016

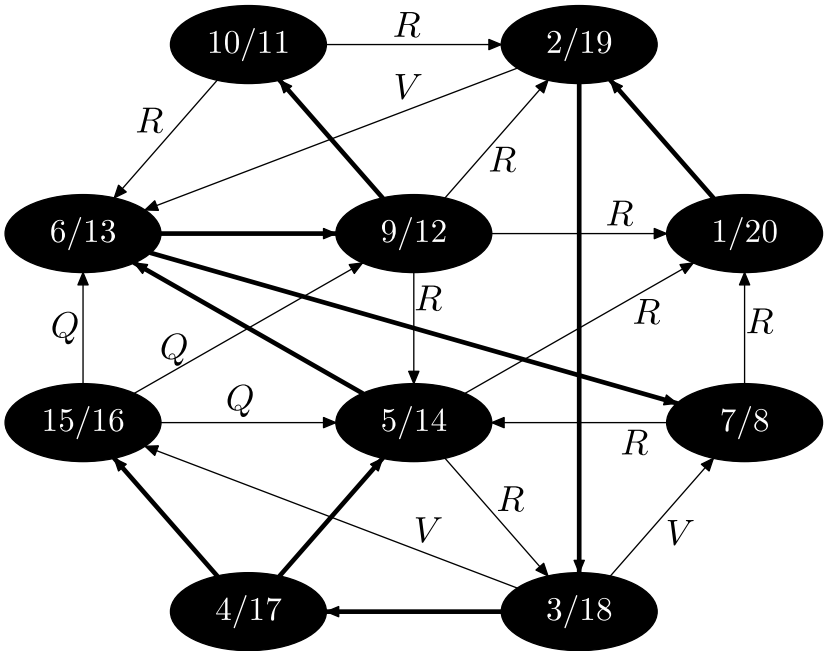
Übung zur Vorlesung Datenstrukturen und Algorithmen

Aufgabe T23

Da der Graph ungerichtet ist, können Anfragen zur Adjazenz zweier Knoten i, j immer auf den Fall $i < j$ zurückgeführt werden. Den Fall $i = j$ beantworten wir immer mit "nein". Dieses entspricht genau der Darstellung in einer Adjazenzmatrix, in der nur die Dreiecksmatrix unterhalb bzw. oberhalb der Diagonalen verwendet werden. Man beachte, daß diese Dreiecksmatrix genau $n(n-1)/2$ Einträge hat, die wir nun geeignet speichern möchten.

Wir verwenden dazu ein Array, in der für die Knoten $1 \leq j \leq n$ in aufsteigender Reihenfolge die entsprechenden Zeilen der unteren Dreiecksmatrix einfach nacheinander gespeichert werden. Wir müssen nur noch effizient herausfinden können, an welcher Stelle im Array die Zeile für den Knoten j zu finden ist. Dazu beobachten wir, daß für den ersten Knoten keine Einträge, für den zweiten Knoten einen Eintrag (Adjazenz zum ersten Knoten), für den dritten Knoten zwei Einträge usw., und für den $j$ten Knoten $j-1$ Einträge gespeichert werden. Für n Knoten ergeben sich also insgesamt $\sum_{j=1}^{n}(j-1) = n(n-1)/2$ nötige Einträge. Außerdem erkennen wir direkt, daß der Beginn der Zeile für den $j$ten Knoten nun an der Position $p(j) := (j-1)(j-2)/2$ im Array zu finden ist. Der der Eintrag zur Adjazenz der Knoten i und j für $i < j$ befindet sich daher an Position $p(j) + i - 1 = (j-1)(j-2)/2 + i - 1$ im Array. Diese Position ist durch zwei Multiplikationen und eine Addition also in konstanter Zeit zu bestimmen.

Aufgabe T24



Der Graph hat eine einzige starke Zusammenhangskomponente.

Aufgabe T25

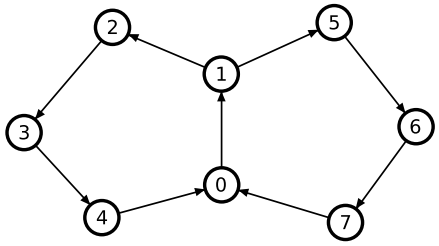
Stellen wir zunächst fest, daß wir uns auf zusammenängende Graphen beschränken können: sollte der Graph mehrere Komponenten besitzen, so kann natürlich nur in einer eine Rückwärtskante auftauchen.

Wir benutzen nun die Tatsache, daß Bäume mit n Knoten genau $n-1$ Kanten besitzen, also $n = m + 1$ gilt. Dies kann man sich einfach veranschaulichen, indem man in einem Baum willkürlich eine Wurzel festlegt und alle Kanten von der Wurzel weg orientiert—auf diese Weise zeigt nun genau eine Kante auf jeden Knoten, abgesehen von der Wurzel, auf welche keine Kante zeigt.

In unserem Tiefensuchbaum gilt nun $n = m$, denn neben den $n-1$ Baumkanten existiert noch die eine Rückwärtskante (es gibt keine Querkanten). Vor diesem Hintergrund ergibt sich die obige Aussage sehr einfach: jeder Tiefensuchbaum muß, da wir ihn als Spannbaum unseres Graphen auffassen können, $n-1$ Baumkanten besitzen. Da der Graph n Kanten besitzt, muß die verbleibende Kante eine Rückwärtskante sein.

Aufgabe H20

Die Aussage ist falsch, wie man sich an folgendem Gegenbeispiel überlegen kann.



Beginnen wir die Tiefensuche am Knoten 1, so findet man genau eine Rückwärtskante, nämlich $(0, 1)$ (egal welchen Kreis man zuerst besucht). Eine Tiefensuche, die am Knoten 0 startet, wird jedoch die Rückwärtskanten $(4, 0)$ und $(7, 0)$ finden!

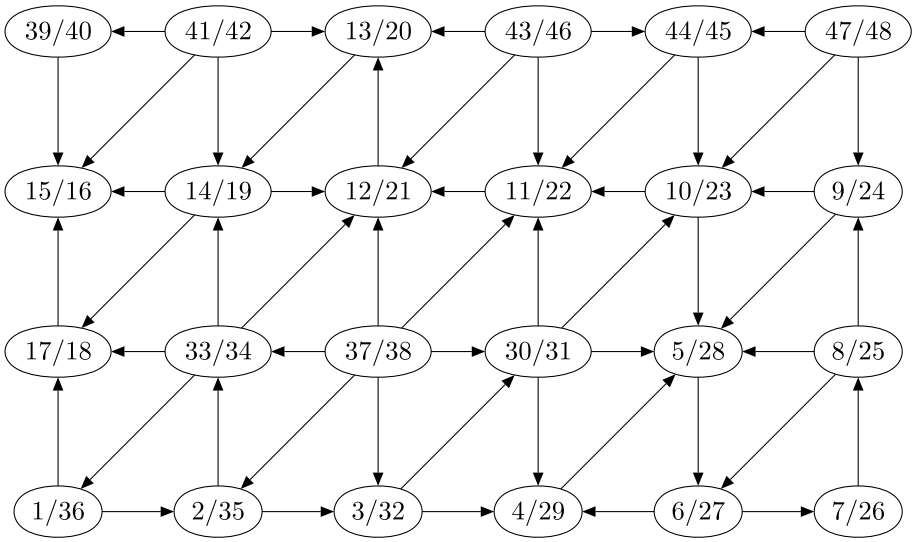
Aufgabe H21

a) Diese Aussage ist wahr. Angenommen Knoten v und u sind in unterschiedlichen Teilbäumen in dem Tiefensuchbaum und Knoten v wurde zuerst besucht. Das bedeutet, dass wir von v nicht zu u weitergegangen sind sondern den Knoten v abgeschlossen haben und backtracking gemacht haben. Dies ist aber nur der Fall wenn es von v keine Kanten mehr gibt die wir noch nicht beachtet haben. Also kann die Kante v, u nicht existieren.

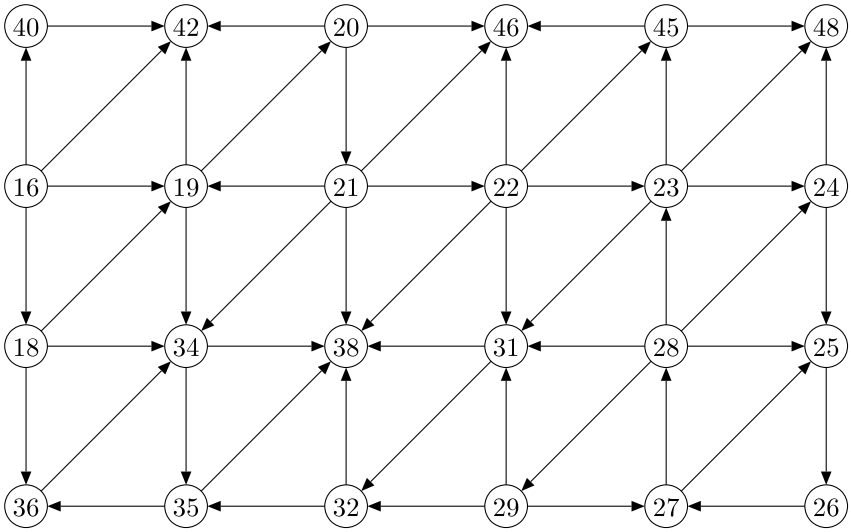
b) Diese Aussage ist Falsch. Querkanten können existieren wenn es die Kante u, v gibt aber nicht v, u und v zuerst besucht wird. Dann wird v abgeschlossen weil es keine weitere Kante gibt die man entlanggehen kann, aber durch die existenz der Kante u, v gibt es eine Querkante.

Aufgabe H22

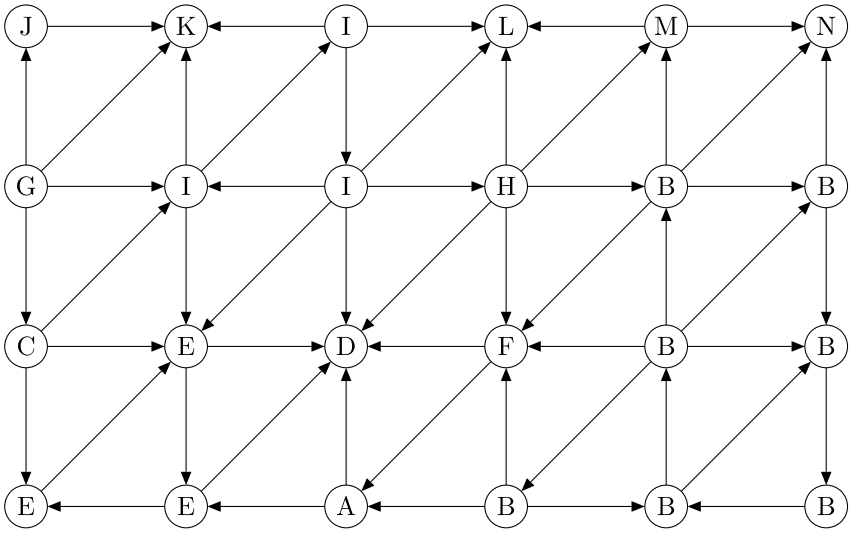
Wir führen zunächst eine Tiefensuche auf dem Graphen aus und erhalten folgendes Ergebnis:



Jetzt drehen wir alle Kanten um. Das ergibt folgenden Graphen. In diesem sind in jedem Knoten die Finish-Zeiten der ursprünglichen Tiefensuche angegeben. Wir werden jetzt auf diesem Graphen eine Tiefensuch in umgekehrter Reihenfolge dieser Zahlen durchführen.



Die Tiefensuche zerfällt in einzelne Tiefensuchen, die jeweils einen Teil des Graphen entdecken. Diese Teile sind die gesuchten starken Zusammenhangskomponenten:


J
K
I
L
M
N
G
I
I
H
B
B
C
E
D
F
B
B
E
E
A
B
B
B