Graphenalgorithmen: Blatt 2

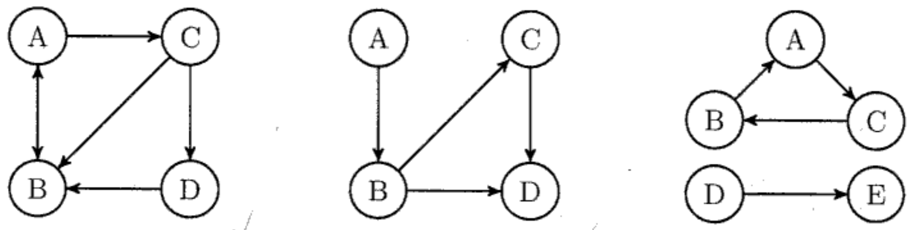
Lukas Kalbertodt, Elena Resch, Mirko Wagner

29. April 2015

A4	A5	Σ
6,5/7	7/8	13,5/15

Aufgabe 4:

(a) (i) Ist zyklisch (`A -> C -> B`) und stark zusammenhängend. (zusammenhängend . ✓
(ii) Ist *weder* zyklisch *noch* stark zusammenhängend, aber zusammenhängend. ✓
(iii) Ist zyklisch (`A -> C -> B`), aber nicht zusammenhängend (also auch nicht stark zusammenhängend). ✓



(i) ✓ (ii) ✓ (iii) ✓

(b) In der $n \times n$ Matrix B mit ist $b_{ii} = \sum_{j=1}^{n} A_{ij} \cdot A_{ji}$. Da A ungerichtet ist, ist $A_{ij} = A_{ji}$. Der Eintrag A_{ij} hat genau dann eine 1, wenn eine Kante von i nach j existiert. Weil $1^2 = 1$ und $0^2 = 0$, sind alle Summanden entweder 0 oder 1. Die Einsen (d.h. die Kanten von Knoten i) werden aufaddiert, also gezählt, und somit ergibt der Eintrag b_{ii}den Knotengrad von i. ✓

(c) Wenn sich zwei Knoten e_1 und e_2 in der selben starken Zusammenhangskomponente befindet, gibt es einen gerichteten Weg von e_1 zu e_2 und umgekehrt. Das heißt, dass in dem transitiven Abschluss des Graphen, eine direkte Kante von e_1 nach e_2 und von e_2 nach e_1 existiert. Die Formel zur Berrechnung von b_{ii} aus (b) lautet: $b_{ii} = \sum_{j=1}^{n} A_{ij} \cdot A_{ji}$. Allerdings ist der transitive Abschluss gerichtet, d.h. A_{ij} und A_{ji} müssen nicht gleich sein. Ein Summand ist nur genau dann 1, wenn es eine Kante von i nach j und umgekehrt gibt. Wie oben bereits gesagt ist das genau dann der Fall, wenn i und j im Ursprungsgraphen in der selben starken Zusammenhangskomponente waren. Mit der Summe zählen wir die Einsen, also die Anzahl der Knoten in der selben starken Zusammenhangskomponente. $-\frac{1}{2}$

Es bleibt zu zeigen, dass für alle k, j mit $k \neq j \neq i$ aus der starken ZHK ein Weg von k nach j existiert

Aufgabe 5:

−1

(a) *Adjazenzmatrix*: Es muss ein Element der Matrix überprüft werden. → $\mathcal{O}(1)$ richtig, aber welches ist das? In dem Array?

Adjazentlisten: Es wird bei dem kleineren Knoten seine Adjazentliste nach dem größeren Knoten durchsucht. Im Worstcase könnte die Liste $|V|$ Einträge halten, die man alle nach dem größeren Knoten durchsuchen muss. → $\mathcal{O}(|V|)$ ✓

(b) Da jeder Knoten n Nachbarn haben könnte, dauert es also mindestens $\Omega(|V|)$.

Adjazenzmatrix: Es muss eine komplette Spalte oder Zeile (in $\mathcal{O}$ Notation gleichwertig) durchgegangen werden. Auch wenn der Knoten nur wenige oder keine Nachbarn hat, muss man trotzdem immer alle $\mathcal{O}(|V|)$ Einträge durchsuchen. → $\mathcal{O}(|V|)$ ✓ Spalte so nicht mehr vorhanden durch Komprimierung

Adjazentlisten: Man fügt zuerst alle Einträge in der Adjazentliste des betreffenden Knoten zu seiner Ergebnismenge hinzu. Danach muss in den Listen aller kleinerer Knoten ($\mathcal{O}(|V|)$) nach dem betreffenden Knoten gesucht werden ($O(|V|)$). → $\mathcal{O}(|V|^2)$

richtig, lässt sich aber auf die Anzahl der Kanten abschätzen: $O(|E|)$