DAP2 – Heimübung 12

Ausgabedatum: 30.06.17 — Abgabedatum: Montag 10.07. 12 Uhr

Schreiben Sie unbedingt immer Ihren **vollständigen Namen**, **Ihre Matrikelnummer** und **Ihre Gruppennummer** auf Ihre Abgaben!

Aufgabe 12.1 (5 Punkte): (Graphen)

Ein ungerichteter Graph $G = (V, E)$ heißt **zusammenhängend**, wenn man von jedem Knoten $u \in V$ aus jeden anderen Knoten $v \in V$ erreichen kann. Sei $|V| = n$.

a) Es gilt folgende Aussage: ein zusammenhängender Graph ohne Schleifen (d.h. ohne Kanten der Form (v, v) für $v \in V$) hat mindestens $n - 1$ Kanten. Argumentieren Sie, warum folgender Induktionsbeweisversuch falsch ist.

"**I.A.** Für $n = 1$ gilt die Aussage, da $1 - 1 = 0$ Kanten immer vorhanden sind. **I.V.** Jeder zusammenhängende Graph mit $n - 1$ Knoten enthält mindestens $n - 2$ Kanten. **I.S.** Aus einem Graphen mit n Knoten entfernen wir einen beliebigen Knoten, der Restgraph hat nach Induktionsvoraussetzung $n - 2$ Kanten. Da der entfernte Knoten mit den anderen $n - 1$ Knoten verbunden sein muss, damit der Graph zusammenhängend ist, enthält der Graph mindestens $n - 1$ Kanten."

b) Wie viele Kanten hat ein zusammenhängender Graph ohne Schleifen höchstens? Begründen Sie Ihre Antwort.

c) Wie viele Kanten hat ein ungerichteter Graph $G = (V, E)$ ohne Schleifen höchstens, wenn er **nicht** zusammenhängend ist? Begründen Sie Ihre Antwort.

Lösung:

a) Wenn wir einen beliebigen Knoten entfernen, ist der Restgraph nicht mehr unbedingt zusammenhängend! Es ist möglich, dass die Entfernung des Knotens den Graphen in mehrere Zusammenhangskompontenten zerfallen lässt.

Korrekterweise sollte man im I.S. argumentieren, wie folgt: wir entfernen einen Knoten und erhalten eine Menge von k Zusammenhangskomponenten (mindestens eine, höchstens $n-1$). Hieraus folgt, dass v mit jeder dieser Komponenten verbunden gewesen sein muss, also einen Grad von mindestens k haben muss. Sei n_i die Anzahl der Knoten in der i-ten Zusammenhangskomponente. Dann gilt

$$\sum_{i=1}^{k} n_i = n - 1 \ ,$$

und da jede dieser Komponenten höchstens $n-1$ Knoten hat, gilt die I.V. für jede dieser Komponenten und wir erhalten als untere Schranke für die Anzahl der Kanten

$$\#Kanten \geq \sum_{i=1}^{k}(n_i - 1) + k = \sum_{i=1}^{k}(n_i) - k + k = n - 1 \ .$$

b) Ein zusammenhängender ungerichteter Graph hat höchstens $(n^2 - n)/2$ Kanten.

Die Aussage lässt sich wie folgt begründen: wir zählen die Kanten in einem vollständigen Graphen, indem wir uns die möglichen Kantentupel ansehen und zählen, wie vielen Kanten sie entsprechen. Es gibt n^2 Tupel der Form (u, v) mit $u, v \in V$, von denen n Tupel die Form (u, u) haben und deshalb wegfallen. Da es sich um einen ungerichteten Graphen handelt, sind die Kanten (u, v) und (v, u) gleich, so dass sich die Anzahl noch einmal halbiert.

Alternativ kann man sich auch überlegen, dass man $n-1$ Kanten braucht, um den ersten Knoten mit allen folgenden zu verbinden, $n-2$ Kanten um den zweiten mit allen Knoten mit höherem Index zu verbinden, und im Allgemeinen $n - i$ Kanten, um den i. Knoten mit allen Knoten mit höherem Index zu verbinden. Das ergibt dann:

$$\sum_{i=1}^{n} n - i = \sum_{i=0}^{n-1} i = \frac{(n-1)n}{2} = \frac{n^2 - n}{2}.$$

Schlussendlich wäre auch für diese Aussage ein formaler Beweis mit Induktion durchaus angebracht. **I.A.** Man beginnt wieder mit $n = 1$, und hat in einem Graphen ohne Schleifen keine Kanten, so dass die Aussage, dass es höchstens $n(n-1)/2 = 0$ sind, erfüllt ist. **I.V.** Jeder Graph mit n Knoten enthält höchstens $n(n-1)/2$ Kanten, für ein festes aber beliebiges n (egal, ob der Graph zusammenhängend ist oder nicht). **I.S.** Wir zeigen dann die Aussage für $n+1$ Knoten: Sei G ein beliebiger Graph mit $n+1$ Knoten. Entfernen wir einen beliebigen Knoten, so gehen dadurch maximal n Kanten verloren, da der Knoten nur mit n anderen Knoten verbunden sein kann. Der verbleibende Graph hat maximal $n(n-1)/2$ Kanten nach Induktionsvoraussetzung. Insgesamt haben wir also maximal $n + n(n-1)/2 = (n^2 + n)/2 = (n+1)n/2$ Knoten wie gewünscht.

c) Ist der Graph nicht zusammenhängend, so erhalten wir die meisten Kanten, wenn wir einen einzelnen Knoten und einen vollständigen Graphen mit $n-1$ Knoten kombinieren. Der vollständige Graph mit $n-1$ Knoten hat laut Aufgabenteil b) maximal $((n-1)^2 - n + 1)/2$ Kanten.

Zusätzliche Argumentation dafür, dass das optimal ist: Eine maximale Lösung besteht aus zwei Zusammenhangskomponenten. Sonst kann man einige der Zusammenhangskomponenten durch zusätzliche Kanten verbinden, ohne dass der gesamte Graph zusammenhängend wird. Seien nun c und $n - c$ die Größen der zwei Zusammenhangskomponenten in einem Graphen. In Teilfaufgabe b) argumentierten wir bereits, dass ein vollständiger Graph mit n Knoten höchstens $(n^2 - n)/2$ Kanten hat. Diesen Wert übernehmen wir nun für die maximale Anzahl der Kanten in jeder Zusammenhangskomponente und erhalten für die maximale Anzahl an Kanten in diesem Graphen

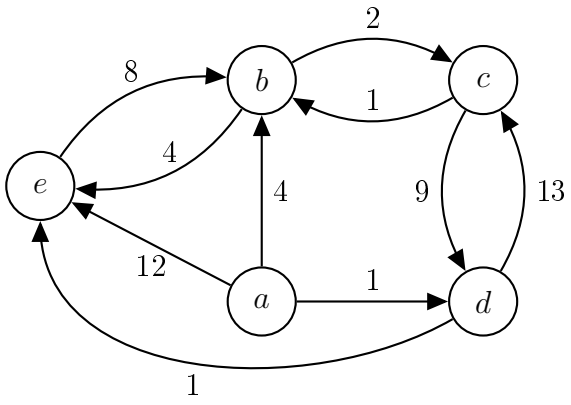
$$\frac{(n-c)^2 - (n-c) + c^2 - c}{2} = \frac{2c^2 + n^2 - 2nc - n}{2} \ .$$

Als Funktion von c ist dies eine nach oben geöffnete Parabel (Polynom zweiten Grades) mit Tiefpunkt bei $c = (2n+1)/4$ und Randmaxima bei $c = 1$ und $c = n-1$. die Randmaxima sind dadurch vorgegeben, dass zwei Zusammenhangskomponenten existieren müssen, und dieser beiden Komponenten mindestens einen Knoten enthalten muss. Setzt man $c = 1$ ein, erhält man genau den oben genannten Wert von $((n-1)^2 - n + 1)/2$ Kanten.

Alternativ stellt man sich nun die zwei Zusammenhangskomponenten der Größen c und $n - c$ vor. Da keine Kante aus der einen in die andere Komponente existieren darf (sonst ist der Graph zusammenhängend), verliert man für jeden der c Knoten in der ersten Komponente $n-c$ Kanten, die nicht vorhanden sein dürfen, also insgesamt $c(n-c)$ Kanten. Will man diesen Term in Abhängigkeit von c minimieren, findet man die Randminima bei $c = 1$ und $c = n - 1$. Eingesetzt in $(n^2 - n)/2 - c(n - c)$, also der maximal möglichen Anzahl von Kanten für einen vollständigen Graphen minus der Anzahl der Kanten, die man verliert, ergibt sich wieder der Term $(n^2 - 3n + 2)/2$, welcher gleich dem oben angegeben Term ist.

Aufgabe 12.2 (5 Punkte): (Algorithmus von Dijkstra)

a) Die Eingabe besteht aus dem folgenden Graphen.



Führen Sie jetzt Algorithmus von Dijkstras ausgehend von a als Startknoten aus und füllen Sie dabei folgendes Schema aus, solange weitere Iterationen benötigt werden:

Nach dem 0. Durchlauf (Initialisierung)

	a	b	c	d	e
d	0	∞	∞	∞	∞
color	w	w	w	w	w

Q: (a,0),(b,∞),(c,∞),(d,∞),(e,∞)

nach dem 1. Durchlauf

	a	b	c	d	e
d					
color					

Q:

usw. Beachten Sie dabei:

i) Geben Sie die Werte nach jedem Durchlauf der while-Schleife im Pseudocode aus der Vorlesung an. Füllen Sie das Schema solange aus, bis alle Knoten schwarz sind.

ii) Verwenden Sie als Prioritätsschlange Q eine sortierte Liste, in der Tupel der Form
(Knoten,Distanzwert)
gespeichert sind, sortiert nach dem Distanzwert.
Geben Sie den Inhalt von Q jeweils unter der Tabelle an.

iii) Die Farben dürfen mit s und w abgekürzt werden.

b) In dem Aufgabenteil a) haben Sie Dijkstras Algorithmus ausgeführt und dabei als Prioritätsschlange eine sortierte Liste verwendet. Erklären Sie, welchen Nachteil bezüglich Laufzeit die Wahl dieser Prioritätsschlange hat, im Vergleich zur der Prioritätschlange aus der Vorlesung.

Lösung:

a) Benötigt werden fünf Durchläufe der While-Schleife, und die entsprechenden Tabellen sehen so aus:

Nach dem 0. Durchlauf (Initialisierung)

	a	b	c	d	e
d	0	∞	∞	∞	∞
color	w	w	w	w	w

Q: (a,0),(b,∞),(c,∞),(d,∞),(e,∞)

nach dem 1. Durchlauf

	a	b	c	d	e
d	0	4	∞	1	12
color	s	w	w	w	w

Q: (d,1),(b,4),(e,12),(c,∞)

nach dem 2. Durchlauf

	a	b	c	d	e
d	0	4	14	1	2
color	s	w	w	s	w

Q: (e,2),(b,4),(c,14)

nach dem 3. Durchlauf

	a	b	c	d	e
d	0	4	14	1	2
color	s	w	w	s	s

Q: (b,4),(c,14)

nach dem 4. Durchlauf

	a	b	c	d	e
d	0	4	6	1	2
color	s	s	w	s	s

Q: (c,6)

nach dem 5. Durchlauf

	a	b	c	d	e
d	0	4	6	1	2
color	s	s	s	s	s

Q:

b) Um die Sortierung aufrecht zu erhalten, muss bei jeder DecreaseKey-Operation der betroffene Knoten neu einsortiert werden. In einer sortierten Liste mit bis zu n Elementen kann diese Operation bis zu $\Omega(n)$ Zeit in Anspruch nehmen. Die Laufzeit von Dijkstras Algorithmus steigt damit auf $\Theta((n+m)\cdot n)$ im schlechtesten Fall (im Vergleich zur Implementierung aus der Vorlesung mit $\Theta((n+m)\cdot \log n)$).