Sarah Ertel
Patrick Greher
Eugen Ljavin

1	2	3	4	Σ

Übungsblatt Nr. 10

(Abgabetermin 05.07.2018)

Aufgabe 1

a)

Für jeden Knoten v in G gilt $1 \leq deg(v) \leq n-1$.
Es müssen also $n-1$ Grade auf n Knoten verteilen werden, womit zwei Knoten den gleichen Grad haben müssen.

b)

Da eine Kante immer zwei Knoten miteinender verbindet, entspricht die Summe aller Grade in G zwei mal m: $\sum_{v \in V} deg(v) = 2 \cdot m$.
Sei V_g die Menge aller Knoten mit geradem Grad und V_u die menge aller Knoten mit ungeradem Grad.
Es muss dann gelten: $2 \cdot m = \sum_{v \in V} deg(v) = \sum_{v_g \in V_g} deg(v_g) + \sum_{v_u \in V_u} deg(v_u)$
Da $\sum_{v_g \in V_g} deg(v_g)$ ein gerades Ergebnis liefert (alle Summanden sind gerade) muss auch $\sum_{v_u \in V_u} deg(v_u)$ ein gerades Ergebnis liefern, damit obige Gleichung erfüllt ist.
$\sum_{v_u \in V_u} deg(v_u)$ kann nur dann ein gerades Egebnis liefern, wenn $|V_u|$ ebenfalls gerade ist.
Somit ist die Anzahl der Knoten in G mit ungeradem Grad gerade.

c)

Die Anzahl an Kanten ist $m = \frac{n \cdot (n-1)}{2} = \frac{n^2 - n}{2}$

d)

Ein d-regulärer Graph muss $n = \frac{2m}{d}$ Knoten haben.
$m = \frac{d \cdot m}{2}$ nach n umgestellt ergibt $n = \frac{2m}{d}$.

Aufgabe 2

a)

Adjazenzmatrix:

	a	b	c	d	e	f	g	h	i
a	0	1	1	1	0	0	0	0	0
b	0	0	0	1	1	0	0	0	0
c	0	0	0	0	1	1	0	0	0
d	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	1	1	1
f	0	0	0	0	0	0	0	0	1
g	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	0

Adjazenzliste:

a	→	b	→	c	→	d
b	→	d	→	e		
c	→	e	→	f		
d						
e	→	g	→	h	→	i
f	→	i				
g						
h						
i						

b)

Lösung: a,b,c,d,e,f,g,h,i
Wähle jeweils einen Knoten, der keine Eingangskante hat und füge ihn zu der SortiertenListe hinzu. Anschließend lösche den Knoten und seine ausgehenden Kanten aus dem Graphen. Wiederhole dieses Prinzip bis kein Knoten mehr im Graph vorhanden ist.

c)

Wähle die erste Zeile in der Matrix. Gehe durch jede Zelle und prüfe ob der Eintrag 1 ist. Falls ja, gehe in den Graphen G' und füge falls noch nicht vorhanden das Element der Zeile ein und erzeuge in G' eine Verbindung von dem Element der Spalte zu dem Element der aktuellen Zeile.
Sobald die Zeile durchlaufen wurde, gehe in die nächste Zeile und wiederhole das gesamte Vorgehen.
Da es sich dabei um zwei verschachtelte for-Schleifen handelt, die die Länge V haben, beträgt der Aufwand $\mathcal{O}(|V|^2)$

d)

Betrachte das erste Element in der Addjazenzliste und füge es falls noch nicht vorhanden in G' ein. Besitzt es Nachfolgeelemente, dann füge diese Ebenfalls in G' ein und erzeuge eine Verbindung von den Nachfolgeelementen in Richtung Element. Anschließend gehe zum nächsten Element und wiederhole den vorherigen Schritt, bis die Liste zuende ist.
Die Laufzeit beträgt hier $\mathcal{O}(|V| + |E|)$ um alle Elemente der Liste zu betrachten. Hinzukommt bei jedem Element die Anzahl der Kanten / Nachfolgerelemente die betrachtet werden.

Aufgabe 3

	S	a	b	c	d	e	f	t
S	0S	4S	6S	∞	∞	∞	∞	∞
a		4S	6S	10a	∞	∞	∞	∞
b			6S	8b	11b	∞	∞	∞
c				8b	10c	15c	∞	∞
d					10c	15c	11d	∞
f						15c	11d	20f
e						15c		20f
t								20f

Aufgabe 4

a)

Die Zuverlässigkeit eines Pfades zwischen s und t mit $s = v_0 \to v_1 \to ... \to v_k = t$ entspricht dem Produkt aller Zuverlässigkeiten der Kanten $\prod_{i=0}^{k} p_i$ wobei p_i die Zuverlässigkeit der Kante v_i ist.

b)

Hat eine Kante die Zuverlässigkeit 1 (das Packet wird mit einer Wahrscheinlichkeit von 100% übertragen), so beträgt die Ausfallwahrscheinlichkeit $1-1=0$ (das Packet wird mit einer Wahrscheinlichkeit von 0% ausfallen). Da das Produkt aller Wahrscheinlichkeiten eines Pfades gebildet wird (siehe 4a), hat der gesamte Pfad, der durch den Knoten mit der Zuverlässigkeit 1 führt, die Ausfallwahrscheinlichkeit von 0, womit dieser der günstigste Pfad ist. Dass dazwischen jedoch hohe Ausfallwahrscheinlichkeiten von z.B. 0.99 liegen können, wird dann nicht mehr berücksichtigt ($0 \cdot x = 0$). In diesem Fall funktioniert der Algorithmus nicht mehr korrekt.

b)

Algorithmus

1. Weise allen Knoten die beiden Eigenschaften "Zuverlässigkeit" und "Vorgänger" zu. Initialisiere die Zuverlässigkeit im Startknoten s mit 1 und in allen anderen Knoten mit ∞

2. Solange es noch unbesuchte Knoten gibt und der Endknoten t noch nicht besucht wurde, wähle darunter denjenigen mit maximaler Zuverlässigkeit aus und

   a) speichere, dass dieser Knoten schon besucht wurde
   b) Berechne für alle noch unbesuchten Nachbarknoten das Produkt des jeweiligen Kantengewichtes (jeweilige Zuverlässigkeit) und der Zuverlässigkeit im aktuellen Knoten
   c) Ist dieser Wert für einen Knoten größer als die dort gespeicherte Zuverlässigkeit, aktualisiere sie und setze den aktuellen Knoten als Vorgänger
3. Alle Vorgänger des Endknotens entsprechen dem zuverlässigsten Pfad

(Beschreibung entnommen aus Wikipedia - Dijkstra-Algorithmus (Informelle Darstellung) und entsprechend dem geforderten Algorithmus der Aufgabenstellung angepasst)

Laufzeit Im worst-case ist der Knoten t der letzte Knoten, der besucht wird und für jeden Knoten muss jeder noch nicht besuchte Nachbarknotn besucht werden. Die Laufzeit beträgt dann $\mathcal{O}(n^2)$ wobei $n = |V|$ ist.

Korrektheit Die Korrektheit des Dijkstra Algorithmus wurde bereits in der Vorlesung bewiesen. Die durchgeführten Änderung haben unter der nachfolgenden Bedingungen keine Auswirkungen auf die Korrektheit des Algorithmus:

1. Die Zuverlässigkeiten sind nicht negativ
2. Die Zuverlässigkeiten können niemals 0 werden (siehe 4b)

Beide Bedingungen werden durch die Aufgabenstellung (für $p(v, w)$ gilt $0 < p(v, w) \leq 1$) garantiert. Der Algorithmus funktioniert folglich korrekt.