FAU, Informatik 2, AUD-Team
aud@i2.cs.fau.de

12. Übung

Abgabe bis 07.02.2018, 10:00 Uhr

Einzelaufgabe 12.1: Sortierverfahren – Theorie – *Fortsetzung* — 11 EP

Betrachten Sie in den folgenden Teilaufgaben die Buchstaben und Sonderzeichen der Zeichenkette „*AuDist>Fun??*“ symbolweise (ohne Anführungszeichen) und in Leserichtung mit Ordnung gemäß ASCII-Tabelle (also z.B. $\$ < A < D < a < u$). Verwenden Sie jeweils die in den Beispielen gezeigte Darstellung! ***Achtung:*** Verwenden Sie ***exakt*** die in der Vorlesung beschriebenen Varianten der Sortierverfahren!

a) Sortieren Sie die Zeichen *aufsteigend durch Zerlegen*. Nach dem Partitionieren führen Sie bitte den rekursiven Abstieg zuerst im Intervall *vor* (L) und dann *nach* (R) dem Pivot und geben Sie auch das Ergebnis nach der Rückkehr aus beiden rekursiven Aufrufen an (Q). Wählen Sie als Pivot stets das letzte Element im Intervall (wie in der Vorlesung). *Beispiel:*

Aktion	*l*				
	F	a	u	l	
L	F	a		l	u
L	F		a	l	u
R	F	a		l	u
Q	F	a		l	u
R	F	a	l		u
Q	F	a	l	u	

b) *Sortieren* Sie die Zeichen *absteigend durch einfaches Fachverteilen*. Legen Sie für jedes Zeichen ein eigenes Fach an, geben Sie in der Tabelle jedoch nur die Fächer mit mindestens einem Zeichen an und stellen Sie ausgelassene Fächer als graue Spalten dar. *Beispiel:*

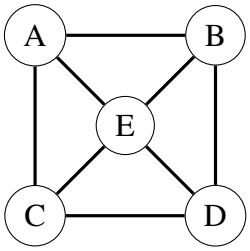
l (Eingabe)								
F, a, l, l, e								
Buckets (Sortiervorgang)								
	F		a		e		l	
	F		a		e		l l	
l (Ergebnis)								
l, l, e, a, F								

Geben Sie Ihre Lösung als `SortierenFortsetzung.pdf` über EST ab.

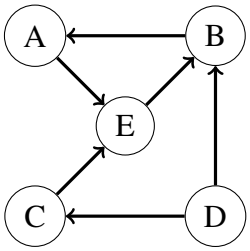
Einzelaufgabe 12.2: Graphkomponenten — 5 EP

Bestimmen Sie für jeden der folgenden Graphen, ob er stark, schwach, nicht zusammenhängend oder nur zusammenhängend ist. Begründen Sie jeweils Ihre Antwort in ein bis zwei Sätzen. Geben Sie Ihre Lösung als `GraphTheorie.pdf` über EST ab.

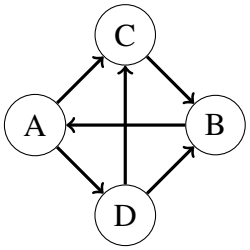
a)



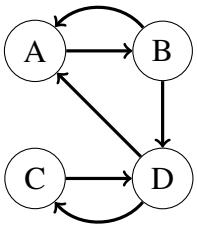
b)



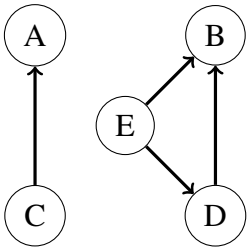
c)



d)

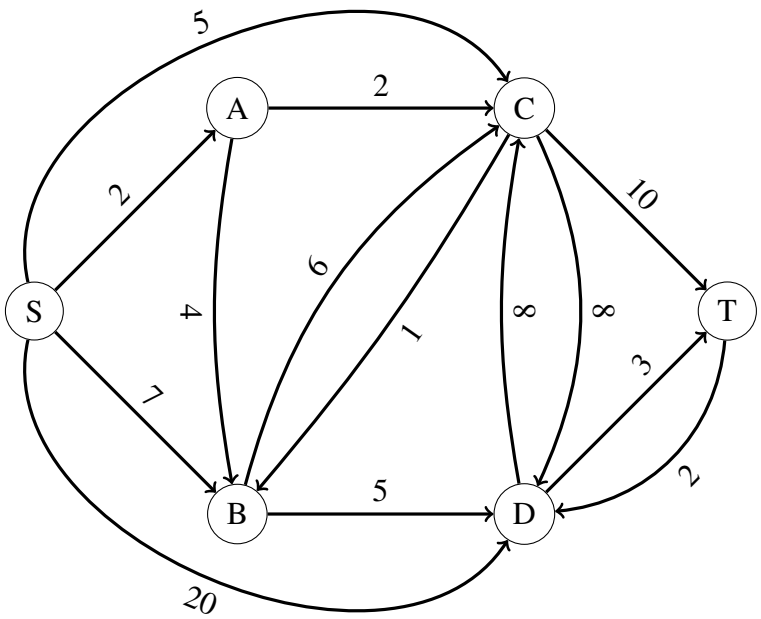


e)



Einzelaufgabe 12.3: Graphen – Theorie — 30 EP

Gegeben sei der folgende gerichtete und gewichtete Graph $\mathcal{G}$:

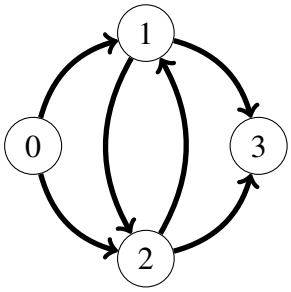


a) Geben Sie die *Adjazenzliste* des Graphen $\mathcal{G}$ nach folgendem Muster an:

$$S \to (X, x) \to (Y, y) \dots$$
$$A \to \dots$$
$$B \to \dots$$
$$\vdots$$
$$T \to \dots$$

b) Geben Sie nun das *Adjazenzfeld* des Graphen $\mathcal{G}$ an, indem Sie die Knoten $\{S, A, B, C, D, T\}$ in dieser Reihenfolge aufsteigend von 0 bis 5 nummerieren. Zur (marginalen) Vereinfachung dürfen Sie die Gewichte weglassen. Beim *Adjazenzfeld* handelt es sich um eine „Array-Einbettung“ eines Graphen, die in dieser Aufgabe wie folgt funktioniert:
Für einen Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ mit $n := |\mathcal{V}|$ Knoten und $e := |\mathcal{E}|$ Kanten wird ein Feld $\mathcal{F}$ der Länge $n+1+e$ angelegt. Die ersten n Einträge (bei Positionen 0 bis $n-1$) repräsentieren die Knoten: Es sind Verweise auf den Bereich ab $n+1$, in dem die Kantenziele verwaltet werden (der Eintrag bei Position n ist ein „Dummy“-Eintrag der auf $n+1+e$, also „hinter“ das Feld verweist, um das Ende des Knotenbereichs zu markieren). Der Ausgangsgrad $outdegree(\nu)$ eines Knotens ν errechnet sich damit direkt aus dem Feld: $outdegree(\nu) = \mathcal{F}[\nu+1] - \mathcal{F}[\nu]$.

Beispiel (Knoten 3 ist eine Senke, daher verweist der Eintrag ebenfalls auf 11.):



0	1	2	3	4	5	6	7	8	9	10	11
5	7	9	11	11	1	2	2	3	1	3	

Das Feld $\mathcal{F}$ ist dann z.B. so zu lesen:
„Die Nachfolger des Knotens 1 befinden sich ab Position 7 aber vor 9 in $\mathcal{F}$ (sind also 2 und 3).“

c) Bestimmen Sie mit dem Algorithmus von *Dijkstra* die kürzesten Pfade ausgehend vom Knoten S zu jedem anderen Knoten in $\mathcal{G}$. Verwenden Sie zur Lösung die unten stehende Tabelle, markieren Sie in jeder Zeile den jeweils als nächstes zu betrachtenden Knoten und führen Sie die *Prioritätswarteschlange* der noch zu betrachtenden Knoten in der letzten Spalte (aufsteigend sortiert nach Priorität). Geben Sie zusätzlich in der letzten Zeile die Länge der gefundenen Pfade zu den einzelnen Knoten erneut an.

S	A	B	C	D	T	*Queue*
0	∞	∞	∞	∞	∞	S
...	...	...	...	...	...	...
			Endergebnis			
...	...	...	...	...	...	—

d) Geben Sie denjenigen kürzesten Pfad vom Knoten S zum Knoten T an, den Sie vorangehend mit dem Algorithmus von *Dijkstra* ermittelt haben.

e) Ermitteln Sie mit Hilfe des Algorithmus von *Floyd* die kürzesten Wege zwischen allen Knotenpaaren in $\mathcal{G}$. Tragen Sie dazu in einer Tabelle folgender Form die jeweils betrachteten Kantenpaare $(u_j, v_i) \wedge (v_i, w_k)$, deren Weglänge $\gamma_{j,i,k} := |(u_j, v_i)| + |(v_i, w_k)|$ sowie die jeweils kürzeste Entfernung $|(u_j, w_k)|$ vor ($\gamma_{j,k}^{alt}$) bzw. nach ($\gamma_{j,k}^{neu}$) einem Schritt ein. Dabei können Sie die Zeilen weglassen, die keinen gültigen Pfad im Graph darstellen. Bearbeiten Sie die Knoten in der Reihenfolge $\{S, A, B, C, D, T\}$ wie beim Adjazenzfeld, sortiert zuerst nach v_i, dann nach u_j und schließlich nach w_k!

Vorgänger u_j	*Knoten* v_i	*Nachfolger* w_k	*„alte Länge“* $\gamma_{j,k}^{alt}$	$\lvert(u_j, v_i)\rvert + \lvert(v_i, w_k)\rvert$ $\gamma_{j,i,k}$	*„neue Länge“* $\gamma_{j,k}^{neu}$
—	S	—	—	—	—
S	A	B	…	…	…
S	A	C	…	…	…
S	B	C	…	…	…
S	B	D	…	…	…
A	B	C	…	…	…
A	B	D	…	…	…
C	B	D	…	…	…
…	…	…	…	…	…

f) Betrachten Sie fortan den Graphen $\mathcal{H}$, der aus $\mathcal{G}$ entsteht, indem alle Kanten aus $\mathcal{G}$ *ungerichtet* übernommen und zusammengelegt werden (d.h. falls es zwischen zwei Knoten in $\mathcal{G}$ zwei entgegengerichtete Kanten $\alpha_\mathcal{G}$ und $\beta_\mathcal{G}$ gibt, dann ist das Gewicht der entsprechende Kante $\phi_\mathcal{H}^{\alpha_\mathcal{G},\beta_\mathcal{G}}$ in $\mathcal{H}$ die Summe der Gewichte von $\alpha_\mathcal{G}$ und $\beta_\mathcal{G}$).
Berechnen Sie den minimalen Spannbaum des ungerichteten Graphen $\mathcal{H}$ und geben Sie die Kanten in derjenigen Reihenfolge an, in der sie der Algorithmus von *Kruskal* in den Ergebnisbaum aufnehmen würde. Geben Sie zusätzlich das Gesamtgewicht des berechneten minimalen Spannbaums an.

g) Betrachten Sie erneut den Graphen $\mathcal{H}$ (wie vorangehend: aus $\mathcal{G}$ abgeleitet, ungerichtet und entgegengerichtete Kanten summiert). Wenden Sie nun aber den Algorithmus von *Prim* an und starten Sie beim Knoten T. Welches Gesamtgewicht hat nun dieser Spannbaum?

Gruppenaufgabe 12.4: Graphalgorithmen — 24 GP

In dieser Aufgabe sollen Sie das Arbeiten mit Graphen in Java üben. Bei der vorgegebenen Implementierung wird ein Graph als Liste seiner Knoten repräsentiert. Implementieren Sie die notwendigen Methoden in den Klassen `GraphAlgorithms` bzw. `GraphIteratorFactory` gemäß ihrer jeweiligen Kommentare in den zugehörigen Schnittstellen und beachten Sie wie immer unbedingt die öffentlichen Testfälle, da sie *wichtiger Bestandteil der Aufgabenbeschreibung* sind!

Viel Erfolg und Tschüss!

Das AuD-Team wünscht Ihnen viel Erfolg in der AuD-Klausur und im weiteren Studium!

46 EP + 24 GP = 70 Punkte