Fachbereich Informatik
Dr. Michael Haupt
Visar Januzaj
Johannes Kinder
Sommersemester 2010

TECHNISCHE UNIVERSITÄT DARMSTADT

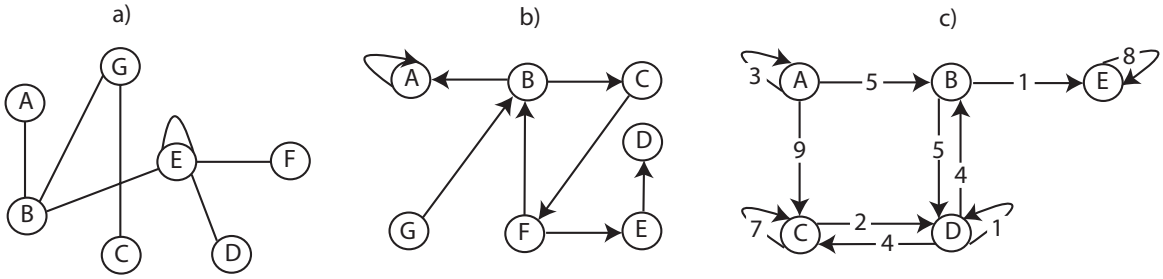
Grundlagen der Informatik 2

Übungsblatt 6

Abgabe: 04.06.2010, 15:00

T 6.1 Graphdarstellungen

Erstellen Sie zu folgenden Graphen jeweils die Adjazenz-, Inzidenz-, und Erreichbarkeitsmatrix.



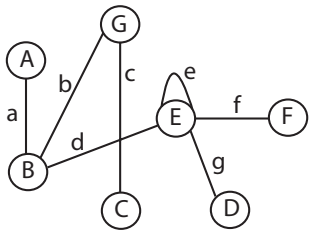
Lösungshinweise

a) Adjazenzmatrix (Knoten alphabetisch sortiert):

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Man beachte, dass die Adjazenzmatrix eines ungerichteten Graphen symmetrisch ist.

Bei der Inzidenzmatrix haben wir eine $|V| \times |E|$-Matrix, wobei jede Zeile einem Knoten entspricht und jede Spalte einer Kante. Da wir hierfür eine Reihenfolge der Kanten benötigen, benennen wir die Kanten alphabetisch mit den Buchstaben a bis g:

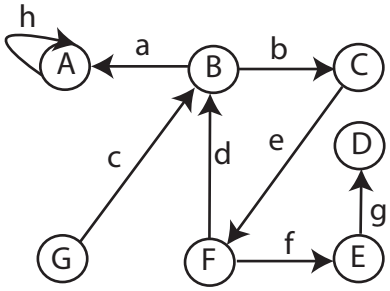


Da der Graph ungerichtet ist, tragen wir für jede inzidente Kante eine 1 ein.

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad E = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Der Graph ist zusammenhängend, daher ist in der Erreichbarkeitsmatrix E jeder Knoten von überall erreichbar.

b) Kantenbenennung:



Inzidenzmatrix:

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

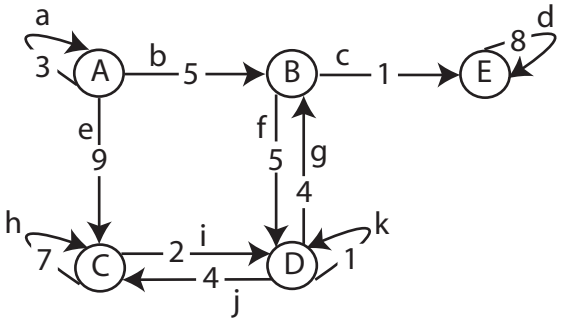
Adjazenzmatrix:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Erreichbarkeitsmatrix:

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

c) Kantenbenennung:



Inzidenzmatrix:

$$I = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Adjazenzmatrix:

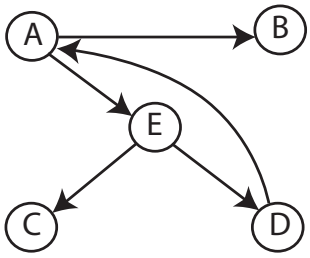
$$A = \begin{pmatrix} 3 & 5 & 9 & 0 & 0 \\ 0 & 0 & 0 & 5 & 1 \\ 0 & 0 & 7 & 2 & 0 \\ 0 & 4 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 8 \end{pmatrix}$$

Erreichbarkeitsmatrix:

$$E = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Für die Erreichbarkeitsmatrix ist die "Entfernung" respektive die Summe der Kantenmarkierungen zu einem anderen Knoten irrelevant. Die Erreichbarkeit wird immer mit einer 1 markiert.

T 6.2 Warshalls Algorithmus



Berechnen Sie für diesen Graphen mit Hilfe des Warshall-Algorithmus die Erreichbarkeitsmatrix (Beachte: "nicht" die transitive Hülle).

Lösungshinweise

Der Warshall-Algorithmus kann vereinfacht so ausgedrückt werden:

Gehe nacheinander alle Knoten $1, \ldots, n$ bzw. $A, B, \ldots$ durch. Betrachte alle Wege der Länge 2, in denen der jeweilige Knoten "Transitknoten" ist, also Bindeglied zwischen zwei anderen Knoten. Verbinde diese Knoten direkt, falls diese Verbindung nicht bereits besteht. Beim vorliegenden Graphen kommen wir damit auf

folgendes Ergebnis:

Initialisierung:

$$A^0 = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Transitknoten A:

$$A^1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Die Transitknoten B und C ändern den Graph und dessen Zwischenmatrix nicht.

Transitknoten D:

$$A^4 = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

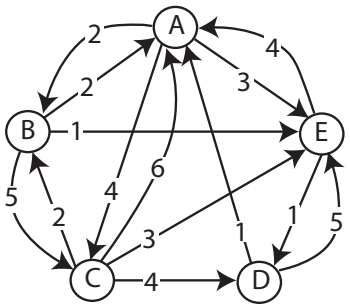
Transitknoten E:

$$A^5 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

T 6.3 Floyd Warshall

In der Vorlesung wurde der Warshall-Algorithmus besprochen, der die Erreichbarkeitsmatrix aus der Adjazenzmatrix berechnet.

a) Floyd-Warshall ist eine Variante des Warshall-Algorithmus, der die Länge der kürzesten Wege zwischen allen Paaren von Knoten eines gewichteten gerichteten Graphen berechnet. Ändern Sie den Warshall-Algorithmus, so dass Sie nun die kürzesten Wege zwischen allen Knotenpaaren bestimmen können.
Hinweis: Bilden Sie die Adjazenzmatrix mit den Kantengewichten als Einträgen.



b) Führen Sie nun Floyd-Warshall für den gegebenen Graphen durch.

Lösungshinweise

a)
```
for (int i = 0; i < nodes.length; i++){
    for (int j = 0; j < nodes.length; j++){
        if (i == j) then D[i,j] = 0;
        else
            if (A[i,j] == 0) D[i,j] = MAX_VALUE;
            else D[i,j] = W[i,j];
    }
}

for (int v = 0; v < nodes.length; v++) { /*Transitknoten ''v''*/
  for (int x = 0; x < nodes.length; x++) { /*von Knoten ''x''*/
    for (int y = 0; y < nodes.length; y++){ /*nach Knoten ''y''*/
        D[x,y] = min(D[x,y], (D[x,v] + D[v,y]));
    }
  }
}
```

b) Der Floyd-Warshall-Algorithmus wird wie folgt durchgeführt:

Initialisierung:

$$D^0 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 5 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ 4 & \infty & \infty & 1 & 0 \end{pmatrix}$$

Transitknoten A:

$$D^1 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 5 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 4 & 6 & 8 & 1 & 0 \end{pmatrix}$$

Transitknoten B:

$$D^2 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 5 & \infty & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 4 & 6 & 8 & 1 & 0 \end{pmatrix}$$

Transitknoten C:

$$D^3 = \begin{pmatrix} 0 & 2 & 4 & 8 & 3 \\ 2 & 0 & 5 & 9 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 4 & 6 & 8 & 1 & 0 \end{pmatrix}$$

Transitknoten D:

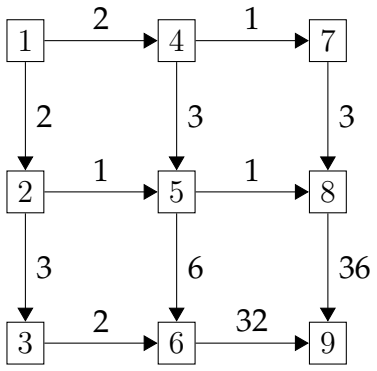
$$D^4 = \begin{pmatrix} 0 & 2 & 4 & 8 & 3 \\ 2 & 0 & 5 & 9 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 2 & 4 & 6 & 1 & 0 \end{pmatrix}$$

Transitknoten E:

$$D^5 = \begin{pmatrix} 0 & 2 & 4 & 4 & 3 \\ 2 & 0 & 5 & 2 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 2 & 4 & 6 & 1 & 0 \end{pmatrix}$$

T 6.4 Linksabbiegen (Zusatzaufgabe)

Der Navigationssystemhersteller TimTim zeichnete das untenstehende Straßennetz eines Stadtteils von Springfield auf, bestehend aus Kreuzungen und Einbahnstraßen. Die Zahlen neben den Straßen geben an, wieviele Minuten man von einer Kreuzung zur nächsten benötigt.



Zunächst übersetzten die Entwickler das Kartenmaterial direkt in einen gewichteten Graphen (Kreuzungen zu Knoten, Straßen und Zeit zu gewichteten Kanten) und verwendeten Dijkstras Algorithmus zur Berechnung der kürzesten Wege.

Bei einem Testlauf stellten sie jedoch fest, dass die geschätzte Zeit für einen Weg stark von der real benötigten Zeit abwich. In diesem Stadtteil sind offenbar keine Ampeln vorhanden, daher nimmt das Linksabbiegen 3 Minuten zusätzlich in Anspruch.

a) Wie können Sie das Straßennetz so in einen Graph übersetzen, dass sich das Problem ohne Veränderung am Dijkstra-Algorithmus lösen lässt?

b) Geben sie die kürzesten Wege ausgehend von Kreuzung 1 als Baum an. Was ist der kürzeste Weg von Kreuzung 1 nach 9 und wie lange dauert die Fahrt?

Lösungshinweise

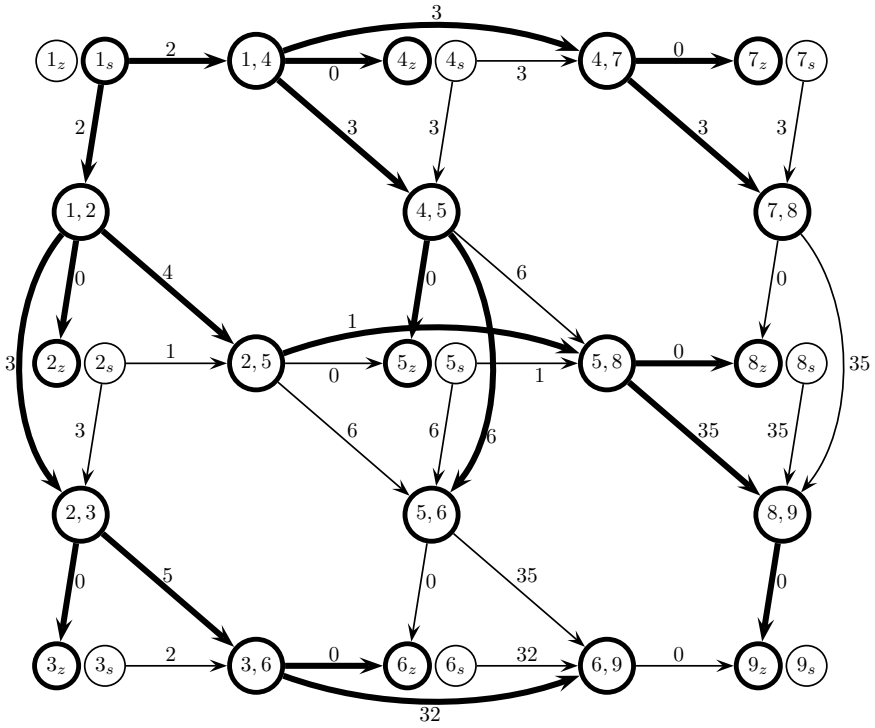
a.) Ausgehend von der ursprünglichen direkten Übersetzung in den Graphen $D = (V, E)$, erstellen wir einen neuen Graphen $D' = (V', E')$:

Für jeden Knoten $v \in V$ werden zwei neue Knoten in V' eingefügt, ein Startknoten s_v und ein Zielknoten z_v. Ebenso wird für jede Kante $(i, j) \in E$ ein weiterer Knoten e_{ij} der Menge V' hinzugefügt.

Bei den Kanten E' sind zwei Knoten e_{ij} und e_{kl} sind genau dann verbunden, wenn $j = k$ gilt. Das dazugehörige Kantengewicht ist dann $c_{jl} + 3$, wenn

links abgebogen wird und ansonsten c_{jl}. Ebenso wird jeder Startknoten s_v mit jedem e_{vi} mit $i \in V$ verbunden und hat die Kantengewichtung c_{vi}. Zuletzt werden alle Knoten e_{ij} mit Zielknoten z_j verbunden. Die Gewichtung dieser Kanten beträgt konstant 0.

Ein möglicher modifizierter Graph für das gegebene Straßennetz ist dann:



b.) Wendet man auf diesen Graphen nun einen Shortest-Path Algorithmus wie Dijkstras Algorithmus an, erhält man als Baum der kürzesten Wege von Kreuzung 1 die im Graphen fett eingezeichneten Kanten.

Um die Länge des Weges von Kreuzung 1 zu Kreuzung 9 zu berechnen muss man nun lediglich den Weg im Kürzeste-Wege-Baum finden und die Kantengewichte addieren:

$$1_s \xrightarrow{2} (1,2) \xrightarrow{4} (2,5) \xrightarrow{1} (5,8) \xrightarrow{35} (8,9) \xrightarrow{0} 9_z$$

Dies ergibt eine Länge $d(9_z) = 42$.

H 6.5 Floyd Warshall (3P)

Die *Inzidenzmatrix* I eines Graphen enthält eine Zeile für jeden Knoten und eine Spalte für jede Kante. Dabei ist $I[i,j] = 1$ wenn Knoten i der Endknoten von Kante j ist, und $I[i,j] = -1$, wenn er der Startknoten ist.

Gegeben ist die Inzidenzmatrix eines gewichteten, gerichteten Graphen mit den Knoten $\{A, B, C, D, E, F, G, H\}$.

$$I = \begin{pmatrix} -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Die Gewichte der Kanten sind gegeben als $(2, 4, 2, 1, 2, 3, 2, 5, 2, 2, 4)$, mit derselben Kantenreihenfolge wie in I.

a) Zeichnen Sie den Graphen zu I mit Kantengewichten ohne Überschneidungen von Kanten.

b) Führen Sie den Floyd-Warshall-Algorithmus auf dem Graphen aus, um die kürzesten Wege zwischen allen Knotenpaaren zu bestimmen. Geben Sie alle Zwischenschritte A^i mit $0 \leq i \leq 8$ an.

H 6.6 Konversion von Graphdarstellungen (3P)

Die *Adjazenzmatrix* A eines Graphen enthält je eine Zeile und eine Spalte für jeden Knoten. Dabei ist im ungewichteten Graphen $A[i, j] = 1$ wenn es eine Kante von Knoten i zu Knoten j gibt.

a) Entwerfen Sie einen Algorithmus in Pseudocode oder Java, um die Adjazenzmatrix eines ungerichteten Graphen in eine Inzidenzmatrix umzuwandeln. *Hinweis:* In ungerichteten Graphen ist $I[i, j] = 1$ wenn i ein Knoten der Kante j ist.

b) Sei A die Adjazenzmatrix eines ungerichteten, ungewichteten Graphen:

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Wenden Sie Ihren Algorithmus auf A an und geben Sie die resultierende Inzidenzmatrix an.

H 6.7 Kneipentour (4P)

Nachdem Sie erfolgreich Ihre GdI 2 Hausaufgaben abgegeben haben, planen Sie eine Kneipentour für Freitag Abend. Unter Ihren Kommilitonen sammeln Sie folgende Vorschläge:

„Von der TU gehen wir zuerst mal ins An Sibin oder in den Ratskeller."
„Vom An Sibin können wir in die Krone oder ins 603qm."
„Nach dem 603qm gehen wir noch zum Herkules."
„Von der Krone gehen wir in den Schlosskeller, ins 603qm oder ins Bruchtal."
„Nach dem Schlosskeller gehen wir noch zum Herkules."
„Vom Ratskeller gehen wir in die Krone oder ins Bruchtal."
„Vom Bruchtal gehen wir noch zum Schlosskeller oder direkt zum Herkules!"

a) Zeichnen Sie einen Graphen mit den möglichen Ortswechseln, die Ihre Kommilitonen vorgeschlagen haben.

b) Sie wollen in dieser Nacht eine möglichst lange Tour machen und beginnend von der TU möglichst viele Orte besuchen. Wie müssen Sie den Graphen umformen, damit Sie den Bellman-Ford-Algorithmus verwenden können, um die längste mögliche Tour zu finden?

   Berechnen Sie eine längste mögliche Tour mit Bellman-Ford. Zeigen Sie die Einzelschritte an einer Tabelle analog zu H4.6.

c) In welcher Reihenfolge müssen Sie (allgemein formuliert) die Kanten des Graphen in jeder Iteration von Bellman-Ford durchlaufen, damit sich möglichst schnell keine Distanzen mehr ändern und Sie den Algorithmus abbrechen können? Wieviele Iterationen werden dann maximal benötigt?

Hinweis: *Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Mit der Abgabe einer Hausaufgabe bestätigen Sie, dass Sie bzw. Ihre Lerngruppe die alleinigen Autoren der Lösungen sind.*