Grundbegriffe der Informatik
WS 2011/12
Tutorium in der Woche 8

Gehalten in den Tutorien Nr. 10, Nr. 14

Philipp Basler (philippbasler@googlemail.com)
Nils Braun (area51.nils@googlemail.com)

KIT - Karlsruher Institut für Technologie

12.12.2011 & 13.12.2011

Inhaltsverzeichnis

Informationen zum nächsten Blatt

Blatt Nr. 8

Abgabetermin	16.12.2011 um 12:30 Uhr
Abgabeort	Briefkasten
Themen	Hübsche Mädchen, Isomorphismen, Bäume, Wegematrizen
Maximale Punkte	20

Häufige Fehler auf dem letzten Übungsblatt

Blatt Nr. 7

- 1. Aufgabe: Beide Richtungen müssen bewiesen werden
- 2. Aufgabe: b) machen wir noch
- 3. Aufgabe: Schreibts doch hin!
- 4. Aufgabe: Nicht vom Spezialfall ausgehen.
- 5. Aufgabe: Genau lesen!

- Zwei unterschiedliche Graphen können das gleiche Aussagen.

Quiz

- Zwei unterschiedliche Graphen können das gleiche Aussagen.

- Zwei unterschiedliche Graphen können das gleiche Aussagen.
- Bei einem ungerichteten Graphen gilt $d^+(x) > d^-(x)$

Quiz

- Zwei unterschiedliche Graphen können das gleiche Aussagen.
- Bei einem ungerichteten Graphen gilt $d^+(x) > d^-(x)$

- Zwei unterschiedliche Graphen können das gleiche Aussagen.
- Bei einem ungerichteten Graphen gilt $d^+(x) > d^-(x)$
- Ein Teilgraph ist definiert mit $V' \subseteq V, E' \subseteq V' \times V'$

- Zwei unterschiedliche Graphen können das gleiche Aussagen.
- Bei einem ungerichteten Graphen gilt $d^+(x) > d^-(x)$
- Ein Teilgraph ist definiert mit $V' \subseteq V, E' \subseteq V' \times V'$

- Zwei unterschiedliche Graphen können das gleiche Aussagen.
- Bei einem ungerichteten Graphen gilt $d^{+}(x) > d^{-}(x)$
- Ein Teilgraph ist definiert mit $V' \subseteq V, E' \subseteq V' \times V'$
- In jedem gerichteten Baum gibt es genau einen Knoten x_0 mit $d^{+}(x_0) = 0 \wedge d^{-}(x_0) \geq 0$

Quiz

- Zwei unterschiedliche Graphen können das gleiche Aussagen.
- Bei einem ungerichteten Graphen gilt $d^+(x) > d^-(x)$
- Ein Teilgraph ist definiert mit $V' \subseteq V, E' \subseteq V' \times V'$
- In jedem gerichteten Baum gibt es genau einen Knoten x_0 mit $d^+(x_0) = 0 \wedge d^-(x_0) \geq 0$

Gegeben sei ein beliebiger Graph G. Wir wollen nun wissen, was es für Verbindungen in diesem Graphen gibt und welche Wege es geben kann.

Gegeben sei ein beliebiger Graph G. Wir wollen nun wissen, was es für Verbindungen in diesem Graphen gibt und welche Wege es geben kann. Hierzu betrachten wir verschiedene Methoden.

Gegeben sei ein beliebiger Graph G. Wir wollen nun wissen, was es für Verbindungen in diesem Graphen gibt und welche Wege es geben kann. Hierzu betrachten wir verschiedene Methoden. Zuerst brauchen wir hierfür eine weitere Defintion

Definition

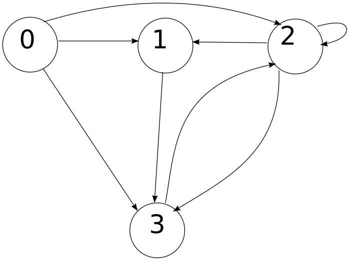
Wir bezeichnen zwei Knoten x und y als *adjazent*, wenn es im betrachteten Graphen durch eine Kante verbunden sind.

Methode 1 : Die Adjazenzliste.

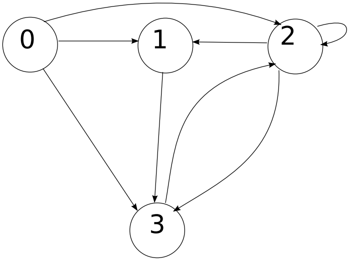
Definition

In der *Adjazenzliste* werden in der zu einem Knoten x alle Knoten eingetragen, die von x direkt erreichbar sind.

Beispiel:



Beispiel:



Für die Adjazentenliste gilt

0	1,2,3
1	3
2	1,2,3
3	2

Methode 2: Die Adjazenzmatrix

Definition

Bei einem Graphen mit n Knoten, bezeichnet die Matrix $A \in \{0,1\}^n \times \{0,1\}^n$ die Adjazenzmatrix des Graphen. Für die Matrix gilt

$$A_{ij} = \begin{cases} 0 & (i,j) \notin E \\ 1 & (i,j) \in E \end{cases}$$

Methode 2: Die Adjazenzmatrix

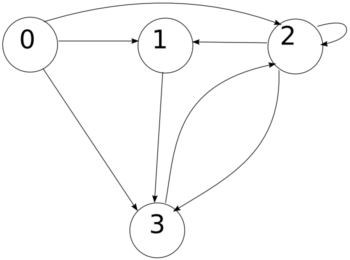
Definition

Bei einem Graphen mit n Knoten, bezeichnet die Matrix $A \in \{0,1\}^n \times \{0,1\}^n$ die Adjazenzmatrix des Graphen. Für die Matrix gilt

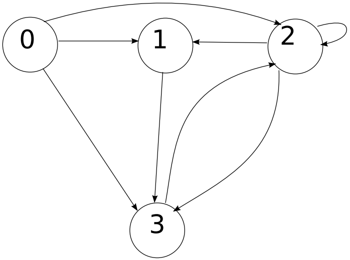
$$A_{ij} = \begin{cases} 0 & (i,j) \notin E \\ 1 & (i,j) \in E \end{cases}$$

Achtung: Bei dieser Definition fangen die Matrixindizes bei 0 an und gehen bis $n-1$.

Beispiel:



Beispiel:



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

Besondere Eigenschaften der Adjazenzmatrix

- Schlinge lässt sich an Wert von A_{ii} erkennen.
- Bei ungerichteten Graphen ist A symmetrisch $\iff A_{ij} = A_{ji}$. Das heißt bei ungerichteten Graphen kann Speicherplatz eingespart werden.

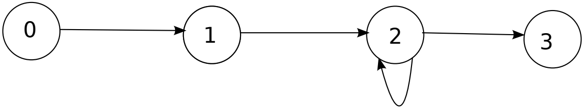
Welche Vorteile haben diese beiden Methoden und wann wird welche bevorzugt?

Welche Vorteile haben diese beiden Methoden und wann wird welche bevorzugt?

- Adjazenzliste braucht wenig Speicherplatz und ist somit geeignet bei dünn besetzten Graphen.
- Adjazenzmatrix braucht immer die gleiche Menge an Speicherplatz (n^2), jedoch bietet sie schnellen Zugriff darauf, ob es eine Kante von i nach j gibt. Insofern besser bei dicht besetzten Graphen.

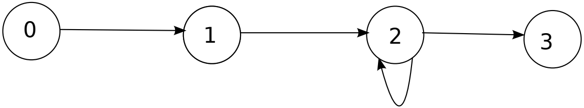
Potenzen einer Adjazenzmatrix

Bestimmen Sie die Matrix A^2 des folgenden Graphen, wobei A die Adjazenzmatrix bezeichne.



Potenzen einer Adjazenzmatrix

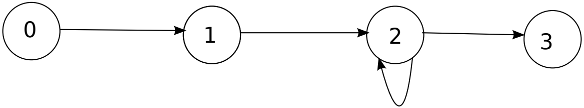
Bestimmen Sie die Matrix A^2 des folgenden Graphen, wobei A die Adjazenzmatrix bezeichne.



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Potenzen einer Adjazenzmatrix

Bestimmen Sie die Matrix A^2 des folgenden Graphen, wobei A die Adjazenzmatrix bezeichne.

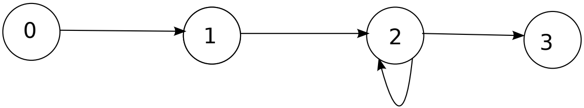


$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$(A^2)_{ij}$ gibt also Auskunft, ob es einen Weg der Länge 2 von i nach j gibt.

Potenzen einer Adjazenzmatrix

Bestimmen Sie die Matrix A^2 des folgenden Graphen, wobei A die Adjazenzmatrix bezeichne.



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$(A^2)_{ij}$ gibt also Auskunft, ob es einen Weg der Länge 2 von i nach j gibt. $\implies$ $(A^n)_{ij}$ gibt also Auskunft, ob es einen Weg der Länge n von i nach j gibt.

Wie kann ich nun also eine schnelle Möglichkeit definieren, ob es *irgendeinen* Weg von i nach j gibt?

Wie kann ich nun also eine schnelle Möglichkeit definieren, ob es *irgendeinen* Weg von i nach j gibt?

Definition

Die Wegematrix W ist definiert als

$$W_{ij} = \begin{cases} 0 & (i,j) \notin E^* \\ 1 & (i,j) \in E^* \end{cases}$$

Sie lässt sich berechnen als

$$W_{ij} = \operatorname{sgn}\left(\left(\sum_{k=0}^{n} A^k\right)_{ij}\right)$$

Wie kann ich nun also eine schnelle Möglichkeit definieren, ob es *irgendeinen* Weg von i nach j gibt?

Definition

Die Wegematrix W ist definiert als

$$W_{ij} = \begin{cases} 0 & (i,j) \notin E^* \\ 1 & (i,j) \in E^* \end{cases}$$

Sie lässt sich berechnen als

$$W_{ij} = \operatorname{sgn}\left(\left(\sum_{k=0}^{n} A^k\right)_{ij}\right)$$

Die Wegematrix ist die Adjazenzmatrix der reflexiv-transitiven Hülle.

Wie kann ich nun also eine schnelle Möglichkeit definieren, ob es *irgendeinen* Weg von i nach j gibt?

Definition

Die Wegematrix W ist definiert als

$$W_{ij} = \begin{cases} 0 & (i,j) \notin E^* \\ 1 & (i,j) \in E^* \end{cases}$$

Sie lässt sich berechnen als

$$W_{ij} = \operatorname{sgn}\left(\left(\sum_{k=0}^{n} A^k\right)_{ij}\right)$$

Die Wegematrix ist die Adjazenzmatrix der reflexiv-transitiven Hülle.
Es folgt also

$$W = A \iff E^* = E$$

Laufzeiten:

Laufzeiten:

- Jede Matrix hat n^2 Einträge, also ergibt sich für die Summe von n Matrizen $n \cdot n^2 = n^3$ Summenoperationen

Laufzeiten:

- Jede Matrix hat n^2 Einträge, also ergibt sich für die Summe von n Matrizen $n \cdot n^2 = n^3$ Summenoperationen
- Es gibt $\sum\limits_{i=0}^{n-1} i = \frac{n(n-1)}{2}$ Matrixmultiplikationen. (Im Algorithmus wird *kein* Speicherplatz für A^i reserviert. Würde bei großen n sehr schnell die Speicherkapazitäten übersteigen.)

Laufzeiten:

- Jede Matrix hat n^2 Einträge, also ergibt sich für die Summe von n Matrizen $n \cdot n^2 = n^3$ Summenoperationen
- Es gibt $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$ Matrixmultiplikationen. (Im Algorithmus wird *kein* Speicherplatz für A^i reserviert. Würde bei großen n sehr schnell die Speicherkapazitäten übersteigen.)
- $(B \cdot C)_{ij} = \sum_{k=0}^{n-1} B_{ik} C_{kj}$, also pro Eintrag einer Matrixmultiplikation , die n^2 Einträge hat, n Multiplikationen und $n - 1$ Additionen.

Laufzeiten:

- Jede Matrix hat n^2 Einträge, also ergibt sich für die Summe von n Matrizen $n \cdot n^2 = n^3$ Summenoperationen
- Es gibt $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$ Matrixmultiplikationen. (Im Algorithmus wird *kein* Speicherplatz für A^i reserviert. Würde bei großen n sehr schnell die Speicherkapazitäten übersteigen.)
- $(B \cdot C)_{ij} = \sum_{k=0}^{n-1} B_{ik} C_{kj}$, also pro Eintrag einer Matrixmultiplikation , die n^2 Einträge hat, n Multiplikationen und $n - 1$ Additionen.
- n^2 Berechnungen der Signum-Funktion

Laufzeiten:

- Jede Matrix hat n^2 Einträge, also ergibt sich für die Summe von n Matrizen $n \cdot n^2 = n^3$ Summenoperationen
- Es gibt $\sum\limits_{i=0}^{n-1} i = \frac{n(n-1)}{2}$ Matrixmultiplikationen. (Im Algorithmus wird *kein* Speicherplatz für A^i reserviert. Würde bei großen n sehr schnell die Speicherkapazitäten übersteigen.)
- $(B \cdot C)_{ij} = \sum\limits_{k=0}^{n-1} B_{ik} C_{kj}$, also pro Eintrag einer Matrixmultiplikation , die n^2 Einträge hat, n Multiplikationen und $n-1$ Additionen.
- n^2 Berechnungen der Signum-Funktion

Insgesamt also

$$n^2 + n^3 + n^2(n + n - 1) \cdot \frac{n(n-1)}{2} = n^5 - \frac{3}{2}n^4 + \frac{3}{2}n^3 + n^2$$

Wir betrachten nur die führende Ordnung. Die Laufzeit des Algorithmus geht also mit n^5

Wir wollens schneller !

Wir wollens schneller !
Benutze bei binären Relationen auf einer Menge M

$$(A \cup B) \circ (C \cup D) = (A \circ C) \cup (A \circ D) \cup (B \circ C) \cup (B \circ D)$$

Wir wollens schneller !
Benutze bei binären Relationen auf einer Menge M

$$(A \cup B) \circ (C \cup D) = (A \circ C) \cup (A \circ D) \cup (B \circ C) \cup (B \circ D)$$

Beweis: Betrachte nun $(A \cup B) \circ E = K$ wobei dann am Ende $E = C \cup D$ gesetzt wird und der Beweis erneut durchgeführt wird.

$$\begin{aligned}
(x,z) \in K &\iff \exists y \in M \ : (x,y) \in A \cup B \land (y,z) \in C \\
&\iff \exists y \in M : ((x,y) \in A \lor (x,y) \in B) \land (y,z) \in C \\
&\iff \exists y \in M : ((x,y) \in A \land (y,z) \in C) \\
&\qquad \lor ((x,y) \in B \land (y,z) \in C) \\
&\iff (x,z) \in A \circ C \lor (x,z) \in B \circ C \\
&\iff (x,z) \in (A \circ C) \cup (B \circ C)
\end{aligned}$$

Betrachten wir nun $F = (Id \cup E)$ für eine Kantenmenge E.

Betrachten wir nun $F = (Id \cup E)$ für eine Kantenmenge E.
So folgt mit dem gerade gezeigten

$$F^2 = (Id \circ E) \cup (Id \circ E) = Id \cup E \cup E^2$$

Betrachten wir nun $F = (Id \cup E)$ für eine Kantenmenge E.
So folgt mit dem gerade gezeigten

$$F^2 = (Id \circ E) \cup (Id \circ E) = Id \cup E \cup E^2$$

Analog :

$$F^4 = (F^2)^2 = (Id \cup E \cup E^2) \circ (Id \cup E \cup E^2) = Id \cup E \cup E^2 \cup E^3 \cup E^4$$

Betrachten wir nun $F = (Id \cup E)$ für eine Kantenmenge E.
So folgt mit dem gerade gezeigten

$$F^2 = (Id \circ E) \cup (Id \circ E) = Id \cup E \cup E^2$$

Analog :

$$F^4 = (F^2)^2 = (Id \cup E \cup E^2) \circ (Id \cup E \cup E^2) = Id \cup E \cup E^2 \cup E^3 \cup E^4$$

Also folgt

$$F^m = \bigcup_{i=0}^{2^m} E^i \qquad m = \lceil \log_2 n \rceil$$

Betrachten wir nun $F = (Id \cup E)$ für eine Kantenmenge E.
So folgt mit dem gerade gezeigten

$$F^2 = (Id \circ E) \cup (Id \circ E) = Id \cup E \cup E^2$$

Analog :

$$F^4 = (F^2)^2 = (Id \cup E \cup E^2) \circ (Id \cup E \cup E^2) = Id \cup E \cup E^2 \cup E^3 \cup E^4$$

Also folgt

$$F^m = \bigcup_{i=0}^{2^m} E^i \qquad m = \lceil \log_2 n \rceil$$

Der Algorithmus sieht nun also folgendermaßen aus

$W \leftarrow A + I$ $\quad //n^2$

$m \leftarrow \lceil \log_2 n \rceil$

for $i \leftarrow 1$ **to** m **do** $\quad //\lceil \log_2 n \rceil$

$\quad W \leftarrow W \cdot W$ $\quad //(n + n - 1) \cdot n$

od

$W \leftarrow \mathrm{sgn}(W)$ $\quad //n^2$

Der Algorithmus sieht nun also folgendermaßen aus

$$
\begin{array}{ll}
W \leftarrow A + I & //n^2 \\
m \leftarrow \lceil \log_2 n \rceil & \\
\textbf{for } i \leftarrow 1 \textbf{ to } m \textbf{ do} & //\lceil \log_2 n \rceil \\
\quad W \leftarrow W \cdot W & //(n + n - 1) \cdot n \\
\textbf{od} & \\
W \leftarrow \operatorname{sgn}(W) & //n^2
\end{array}
$$

Wir brauchen also nurnoch $n^2 + \lceil \log_2 n \rceil \left((2n-1) \cdot n^2\right) + n^2$ Rechenoperationen
Betrachten wir nun $n = 10^{20}$ Einträge. Das Verhältnis von diesem zum ersten Algorithmus beträgt dann $1.34 \cdot 10^{-34}\%$

Nun noch schneller !

Nun noch schneller ! Mit dem Warshall-Algorithmus :

Nun noch schneller ! Mit dem Warshall-Algorithmus :

for $i \leftarrow 0$ **to** $n-1$ **do**

  for $j \leftarrow 0$ **to** $n-1$ **do**

$$W_{ij} \leftarrow \begin{cases} 1 & i = j \\ A_{ij} & i \neq j \end{cases}$$

  od

od

for $k \leftarrow 0$ **to** $n-1$ **do**

  for $i \leftarrow 0$ **to** $n-1$ **do**

    for $j \leftarrow 0$ **to** $n-1$ **do**

$$W_{ij} \leftarrow \max\left(W_{ij}, \min(W_{ik}, W_{kj})\right)$$

    od

  od

od

Nun noch schneller ! Mit dem Warshall-Algorithmus :

for $i \leftarrow 0$ **to** $n-1$ **do** $//n$

  for $j \leftarrow 0$ **to** $n-1$ **do** $//n$

    $W_{ij} \leftarrow \begin{cases} 1 & i = j \\ A_{ij} & i \neq j \end{cases}$ $//1$

  od

od

for $k \leftarrow 0$ **to** $n-1$ **do** $//n$

  for $i \leftarrow 0$ **to** $n-1$ **do** $//n$

    for $j \leftarrow 0$ **to** $n-1$ **do** $//n$

      $W_{ij} \leftarrow \max\left(W_{ij}, \min(W_{ik}, W_{kj})\right)$ $//1$

    od

  od

od

Da wir hier eine Binäre Relation haben gilt

$$\max\left(W_{ij}, \min(W_{ik}, W_{kj})\right) = W_{ij} \vee \left(W_{ik} \wedge W_{kj}\right)$$

Da wir hier eine Binäre Relation haben gilt

$$\max\left(W_{ij}, \min(W_{ik}, W_{kj})\right) = W_{ij} \vee \left(W_{ik} \wedge W_{kj}\right)$$

Es ergibt sich insgesamt also eine Laufzeit von

$$n^3 + n^2$$

Der Warshall-Algorithmus braucht bei $n = 10^5$ im Vergleich zur

Da wir hier eine Binäre Relation haben gilt

$$\max\left(W_{ij}, \min(W_{ik}, W_{kj})\right) = W_{ij} \vee (W_{ik} \wedge W_{kj})$$

Es ergibt sich insgesamt also eine Laufzeit von

$$n^3 + n^2$$

Der Warshall-Algorithmus braucht bei $n = 10^5$ im Vergleich zur

- zweite Variante 74%

Da wir hier eine Binäre Relation haben gilt

$$\max\left(W_{ij}, \min(W_{ik}, W_{kj})\right) = W_{ij} \vee \left(W_{ik} \wedge W_{kj}\right)$$

Es ergibt sich insgesamt also eine Laufzeit von

$$n^3 + n^2$$

Der Warshall-Algorithmus braucht bei $n = 10^5$ im Vergleich zur

- zweite Variante 74%
- ersten Variante 10^{-36}%

Da wir hier eine Binäre Relation haben gilt

$$\max\left(W_{ij}, \min(W_{ik}, W_{kj})\right) = W_{ij} \vee \left(W_{ik} \wedge W_{kj}\right)$$

Es ergibt sich insgesamt also eine Laufzeit von

$$n^3 + n^2$$

Der Warshall-Algorithmus braucht bei $n = 10^5$ im Vergleich zur

- zweite Variante 74%
- ersten Variante 10^{-36}%

an Rechenoperationen

Was sagt die Matrix W an der Stelle W_{ij} nach dem $k-ten$ Durchlauf im Warshall-Algorithmus aus?

Was sagt die Matrix W an der Stelle W_{ij} nach dem $k-ten$ Durchlauf im Warshall-Algorithmus aus?
Stimmt dieser Algorithmus eigentlich?

Was sagt die Matrix W an der Stelle W_{ij} nach dem $k-ten$ Durchlauf im Warshall-Algorithmus aus?
Stimmt dieser Algorithmus eigentlich?

Schleifeninvariante

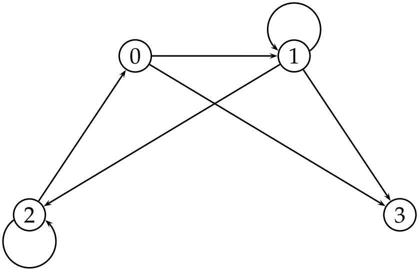
Für alle $i, j \in \mathbb{G}_n \wedge i \neq j$ gilt : Nach k Durchläufen hat die Matrix den Wert 1 an der Stelle i, j, genau dann wenn es einen wiederholungsfreien Pfad von i nach j über Knoten in $\mathbb{G}_k$ gibt. Bei $i = j$ steht dort eine 1. Ansonsten 0.

Schleifeninvariante

Für alle $i, j \in \mathbb{G}_n \land i \neq j$ gilt : Nach k Durchläufen hat die Matrix den Wert 1 an der Stelle i, j, genau dann wenn es einen wiederholungsfreien Pfad von i nach j über Knoten in $\mathbb{G}_k$ gibt. Bei $i = j$ steht dort eine 1. Ansonsten 0.

Beweis auf Seite 117 im GBI-Skript von Herr Worsch.

Gegeben sei folgender Graph G



Geben Sie die Adjazenzliste, die Adjazenzmatrix und die Wegematrix zu diesem Graphen an. Benutzen Sie für die Wegematrix den Warshall-Algorithmus und geben Sie dabei alle Zwischenmatrizen sowie die Initialisierungsmatrix an.

Lösung zu Aufgabe 1

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

Lösung zu Aufgabe 1

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

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

Lösung zu Aufgabe 1

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

$$W = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad W_0 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Lösung zu Aufgabe 1

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

$$W = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad W_0 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$W_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Lösung zu Aufgabe 1

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

$$W = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad W_0 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$W_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \; W_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Lösung zu Aufgabe 1

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

$$W = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad W_0 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$W_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} W_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

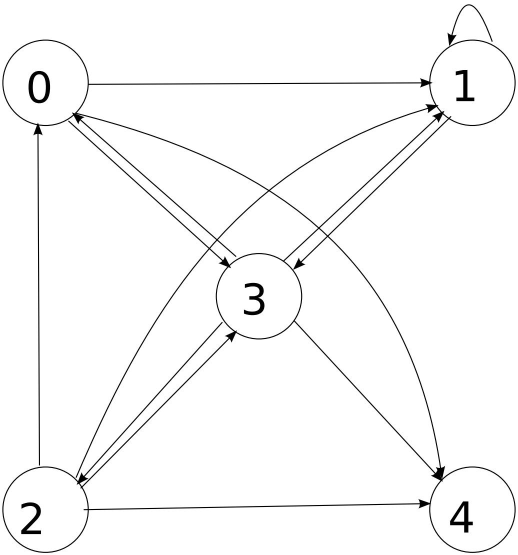
$$W_3 = W_2$$

Gegeben sei folgende Adjazenzmatrix

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

Zeichnen Sie den dazugehörigen Graphen und geben Sie die Initialisierungsmatrix sowie alle Zwischenmatrizen beim Warshall-Algorithmus an.

Lösung zu Aufgabe 2



Lösung zu Aufgabe 2

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

Lösung zu Aufgabe 2

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

$$W_0 = W = W_1 = W_2$$

Lösung zu Aufgabe 2

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

$$W_0 = W = W_1 = W_2$$

$$W_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Lösung zu Aufgabe 2

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

$$W_0 = W = W_1 = W_2$$

$$W_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$W_4 = W_3$$

Was ihr nun wissen solltet

- Wie man eine Adjazenzmatrix aus einem Graphen aufstellt und umgekehrt.
- Wie man die Wegematrix bildet.
- Was die Wegematrix mit der reflexiv-transitiven Hülle zu tun hat.
- Wie ihr Laufzeiten von Code bestimmen könnt.

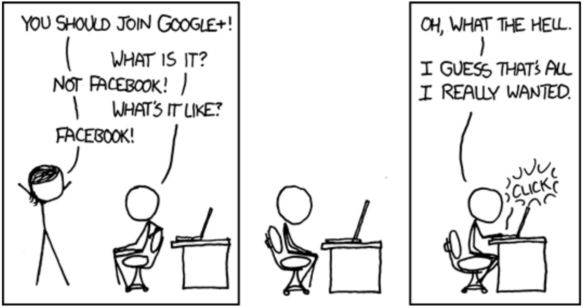


Abbildung: `http://www.xkcd.com`

Kontakt via E-Mail an Philipp Basler oder Nils Braun
gbi.ugroup.hostzi.com