8 Algorithmen in Graphen

8.1 Repräsentation von Graphen im Rechner

Man vergleiche Adjazenzliste und Adjazenzmatrizen:

- Bei den Listen hat man „schnell“ Zugriff auf alle adjazenten Knoten, bei der Matrix muss man alle Knoten überhaupt durchgehen, um zu sehen, welche Nachbarn sind.

- Bei der Matrix kann man „schnell“ herausfinden, ob es eine Kante zwischen zwei Knoten i und j gibt, bei den Listen muss man unter Umständen alle Nachbarn durchgehen.

- Wann spart was Speicherplatz? (Listen bei „relativ wenigen“ Knoten)

Adjazenzmatrizen:

- Woran erkennt man eine Schlinge? (1 auf der Diagonale)

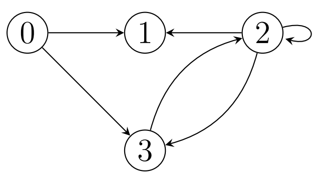
- Beispiele machen, z. B. $A =$ alles Einsen

- welche besondere Eigenschaft haben die Adjazenzmatrizen ungerichteter Graphen? (Symmetrie bzgl. Hauptdiagonale)

- Beispiele von Matrix zum Graphen und zurück

8.1.1 Wegematrix

- erst mal einfach durch Hingucken für einen Graphen eine bestimmen, z. B. für



$$\mathrm{W} = \begin{array}{c} \\ {\scriptstyle 0} \\ {\scriptstyle 1} \\ {\scriptstyle 2} \\ {\scriptstyle 3} \end{array} \begin{array}{c} \begin{array}{cccc} {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 2} & {\scriptstyle 3} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

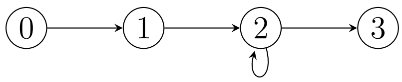
- Wie sieht die Wegematrix aus, wenn $A =$alles Einsen? ($W = A$)

- etwas schwieriger: Wann ist allgemein $W = A$? (Wenn $E^* = E$, also wenn Kantenrelation reflexiv und transitiv)

8.2 Berechnung der 2-Erreichbarkeitsrelation und Rechnen mit Matrizen

8.2.1 Berechnung von E^2

- wie im Skript/auf den VL-Folien noch ein ausführliches Beispiel, vielleicht für



8.2.2 Matrixmultiplikation

- zumindest die Infowirte müssen das üben. Man nehme z. B. die 4×4-Matrizen mit $A_{ij} = \begin{cases} 1 & \text{falls } i \leq j \\ 0 & \text{sonst} \end{cases}$ und $B_{ij} = \begin{cases} 1 & \text{falls } i > j \\ 0 & \text{sonst} \end{cases}$ und berechne AB und BA
- aus der Vorlesung; noch mal durchgehen:

```
for i ← 0 to ℓ − 1 do
  for j ← 0 to m − 1 do
    C_ij ← 0
    for k ← 0 to n − 1 do
      C_ij ← C_ij + A_ik · B_kj
    od
  od
od
```

Einheitsmatrizen

- $\mathrm{I} \cdot A = A$ nachrechnen:

$$(\mathrm{I} \cdot A)_{ij} = \sum_{v=0}^{n-1} \mathrm{I}_{iv} \cdot A_{vj} = \mathrm{I}_{ii} \cdot A_{ij} = A_{ij}$$

8.3 Berechnung der Erreichbarkeitsrelation

8.3.1 quadrierte Adjazenzmatrix

- inhaltliche Bedeutung von $(A^2)_{ij}$ klar machen

8.3.2 Erste Möglichkeit für die Berechnung der Wegematrix

8.3.3 Zählen durchzuführender arithmetischer Operationen

Zählen von Operationen (darauf werden wir aber nächste Woche nochmal genauer kommen):

- so was wie $\sum_{i=1}^{n-1} i = n(n-1)/2$ klar machen
- ich benutze immer die Methode, mit der angeblich Gauß schon als Schulkind auffiel, als alle Schüler $1+2+3+\cdots+100$ ausrechnen sollten: erster + letzter Summand = zweiter + vorletzter Summand = ..., also insgesamt Wert = so eine Zweiersumme mal Anzahl Summanden halbe (funktioniert auch bei ungerader Anzahl Summanden)

8.3.4 Weitere Möglichkeiten für die Berechnung der Wegematrix

Wegematrix schneller (n^4):

- Wenn man eine feste Gesamtzeit zur Verfügung hat und mit dem n^5 Algorithmus gerade noch Probleminstanzen mit $n = 1000$ schafft: Wie große Probleminstanzen schafft man mit dem n^4 Algorithmus?

Von der Rechenregel

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

für Relationen kann man sich mal einen Teil klar machen, z. B. falls alles binäre Relationen auf Menge M sind:

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

(anderer Teil analog) Beweis durch Nachprüfen beider Inklusionen, evtl. gleichzeitig:

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

Allerdings:

- beim dritten $\iff$ braucht man Distributivgesetz für Aussagenlogik: durch Nachdenken klar machen
- beim vierten $\iff$ auch **unbedingt genau nachdenken**: es ist wichtig, dass da ein **Oder** steht
- für $\wedge$ wäre das folgende **FALSCH FALSCH FALSCH** :

$$\exists y \in M : (\mathcal{A}(y) \wedge \mathcal{B}(y)) \iff \exists y \in M : \mathcal{A}(y) \wedge \exists y \in M : \mathcal{B}(y)$$

  HIER IST $\iff$ FALSCH: ES GILT NUR $\Longrightarrow$!

- Gegenbeispiel $\exists y \in \mathbb{N}_0 : y = 1$ und $\exists y \in \mathbb{N}_0 : y = 2 \ldots$
- Wenn Sie sich nicht in der Lage sehen, das klar rüber zu bringen, dann lieber im Beweis umgangssprachlich argumentieren.

Wegematrix schneller ($n^3 \log_2 n$):

- Wenn man eine feste Gesamtzeit zur Verfügung hat und mit dem n^5 Algorithmus gerade noch Probleminstanzen mit $n = 1000$ schafft: Wie große Probleminstanzen schafft man mit dem $n^3 \log_2 n$ Algorithmus?

8.4 Algorithmus von Warshall

- Korrektheit möglichst klar machen, denn der gleiche Trick wird vielleicht noch mal auftauchen bei der Konstruktion von regulären Ausdrücken zu endlichen Automaten.
- auf dem Übungsblatt vor 2 Jahren (Aufgabe 9.1) war eine Aufgabe zum Durchnudeln des Algorithmus: http://gbi.ira.uka.de/archiv/2010/blatt-9-aufgaben.pdf