Vorlesung Mathematik 11.7.18

Mit Hilfe der Adjazenzmatrix eines Digraphen (gerichteten Graphen) und deren Potenzen kann man Aussagen treffen über die Anzahl und die Länge von Wegen in diesem Graphen.

A^r r-te Potenz

Beh: $(a_{ij})^{(r)}$ gibt die Anzahl der verschiedenen Wege (Pfeilfolgen) der Länge r von x_i nach x_j an

Beweis: (vollständige Induktion)

Induktionsanfang: $a_{ij}^{(1)} = 1$ es ex. Weg von x_i nach x_j mit Länge 1

Induktionsvoraussetzung: $(a_{ij})^{(r)} = k$ es ex. k Kantenfolgen der Länge r von x_i nach x_j

Kantenfolgen der Länge $r+1$ von x_i nach x_j entstehen aus den Kantenfolgen von x_i nach x_j, falls sich hier eine Kante einfügen lässt, d.h. Verlängerung von r nach $r+1$

Es wird $A^r \cdot A = A^{r+1}$

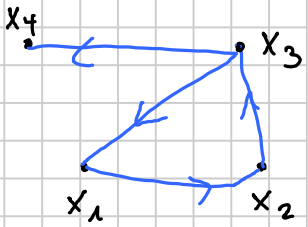
Induktionsschritt: $r \to r+1$

Allgemein: Bei der Matrizenmultiplikation zählen nur Summanden ungleich Null, das ist z.B. der Fall, wenn in A^r an der Stelle $(a_{ij})^{(r)} = k$ steht und eben nur der Summand berücksichtigt wird, der auch bei der Einzelmultiplikation ungleich Null ist.

Ind. schluss: Die Beh. gilt für alle $r \in \mathbb{N}$

Def: Wegematrix $W = (w_{ij})$ mit $w_{ij} = \begin{cases} 1, & \text{falls es einen Weg gibt} \\ 0, & \text{falls es keinen Weg gibt} \end{cases}$

Bp:



$$A = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad A^3 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad A^4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

z.B. es ex ein Weg der Länge 2 von x_1 nach x_3

z.B. es ex ein Weg der Länge 3 von x_2 nach x_2

Wegematrix: $A + A^2 + A^3 + A^4 = \begin{pmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 2 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

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

Ausflug in ein Spezialthema der Graphentheorie: Matching

Matching (Zuordnung, Paarung)

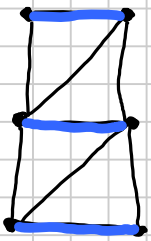
Ein Matching in einem Graphen ist eine Menge von Kanten, die jeweils nur mit einem Knoten inzidieren.

Matching in bipartiten Graphen (zwei disjunkte Knotenmengen und Kantenverbindungen nur zwischen den Knoten der disj. Knotenmengen)

kein Knoten wird zweimal zugeordnet!

Ziel: maximale Zuordnung

nicht erweiterbares maximales Matching



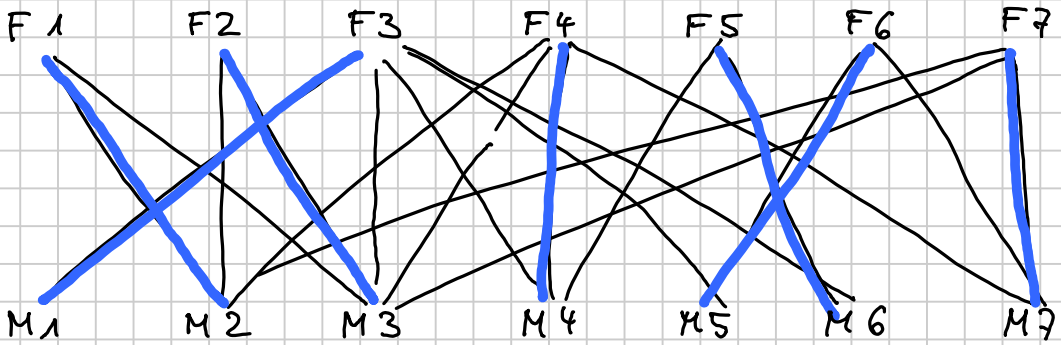
perfektes Matching

Es gilt: 2x Kantenzahl = Knotenzahl

Bp: Heiratsvermittlung

7 Frauen 7 Männer

	Männerwünsche
F1	M2, M3
F2	M2, M3
F3	M1, M3, M4, M5, M6
F4	M2, M3, M4, M7
F5	M4, M6
F6	M5, M7
F7	M2, M3, M7

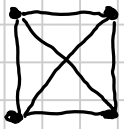


Perfektes Matching

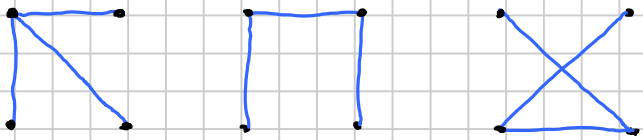
Gerüste in unbewerteten Graphen

Def: Gerüst ist ein Teilgraph eines zusammenhängenden Graphen, der ein Baum ist und genauso viele Knoten wie der Graph selbst besitzt.

Bp:



Gerüste:



insgesamt $\binom{6}{3} = \frac{6!}{3!\cdot(6-3)!} = \frac{4\cdot 5\cdot \cancel{6}^{2}}{1\cdot 2\cdot \cancel{3}^{1}} = 20$ Gerüste

Andere Bezeichnung: Aufspannender Baum
"spanning tree"

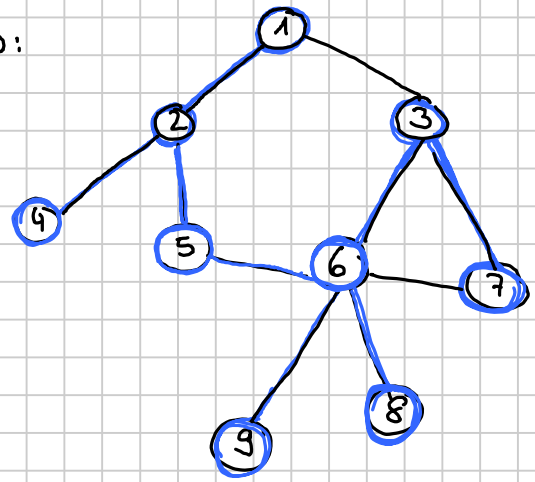
Durchlaufen von Graphen

Viele Graphen, die eine Anwendung repräsentieren, sind sehr groß

Ziel: unnötige Kanten weglassen, dennoch den Zusammenhang erhalten

Suche mit System: Tiefensuche oder Breitensuche

Bp:



Suche Weg, der jeden Knoten einmal benutzt!

Algorithmus: Tiefensuche

1) Startknoten wählen hier ①

2) Besuche einen der möglichen Nachbarn (falls mehrere: einen auswählen (alphabetisch, numerisch ...))

hier ②

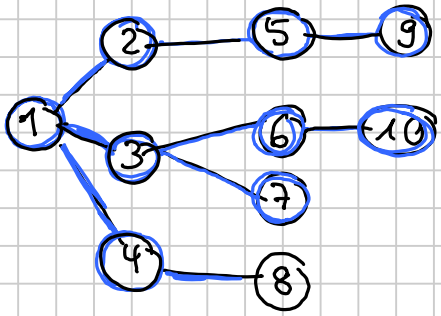
Markiere den Knoten als besucht

3) Besuche von diesem Knoten aus wieder einen Nachbarn hier ④

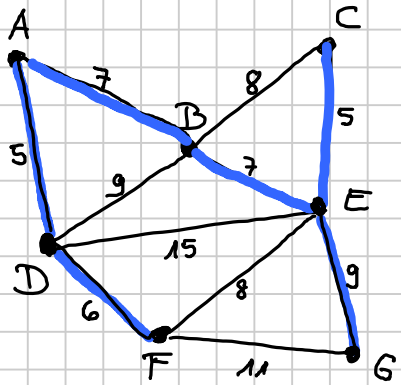
4) Sind bei einem Knoten alle Nachbarn markiert bzw. keine mehr da, so gehe zum vorigen Knoten zurück hier zurück nach ②

Breitensuche: Startknoten festlegen und alle von diesem Startknoten erreichbare Knoten werden markiert usw.

Bp



Aufsuchen eines MST (minimal spanning tree) in einem bewerteten Graphen



Auffinden eines MST durch den Algorithmus von Kruskal (Joseph Kruskal 1956)

(greedy - Algorithmus)

für ungerichtete bewertete Graphen

1) Suche in G (bewertet) die Kante mit dem kleinsten Gewicht

falls sie einen Kreis bildet, dann verwerfen — hier AD mit 5

1) wird so oft wiederholt, bis nichts mehr geht — alternativ: CE mit 5

Bp: Sieben Dörfer, die durch Straßen verbunden werden sollen

	D1	D2	D3	D4	D5	D6	D7
D1	0	12	10	0	14	0	20
D2	12	0	12	10	6	0	0
D3	10	12	0	4	0	0	7
D4	0	10	4	0	0	6	0
D5	14	6	0	0	0	6	8
D6	0	0	0	6	6	0	4
D7	20	0	7	0	8	4	0

Herstellungskosten in TSD-€ für die Verbindungswege

Gesucht wird ein MST, der die minimalen Gesamtherstellungskosten für das geplante Straßennetz darstellt. (mit Alg. v. Kruskal)

Graph möglichst kreuzungsfrei