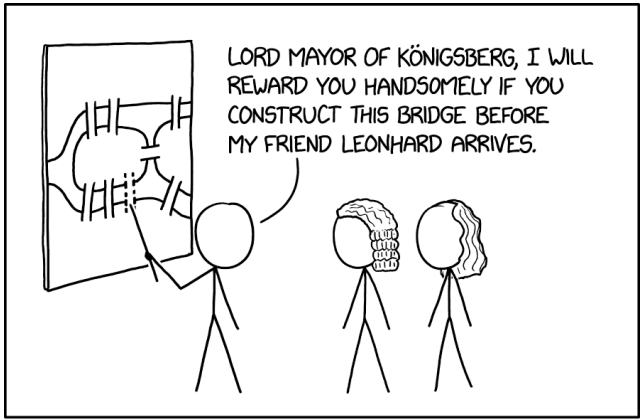
ETH zürich

D&A - Übungsstunde 11

Diese Folien basieren auf denjenigen der Vorlesung, wurden aber durch den Assistenten Adel Gavranović adaptiert und erweitert

Comic der Woche



I TRIED TO USE A TIME MACHINE TO CHEAT ON MY ALGORITHMS FINAL BY PREVENTING GRAPH THEORY FROM BEING INVENTED.

xkcd

Übersicht

Heutiges Programm

Intro
Follow-up
Feedback zu [code]expert
Graphrecap
All-pairs Shortest Path Problem
Live [code]expert
In-Class-Exercise (theoretisch)
Alte Prüfungsfragen (26.1.2018)
Outro

n.ethz.ch/~agavranovic

- Link zum Material für die Übungsstunden
- Webseite des Assistenten
- Mail an Assistenten

1. Intro

Intro

- Tasks 9-13 sind relevant um Bonusaufgabe freizuschalten

2. Follow-up

Follow-up aus vorherigen Übungsstunden

- Um mit einer DFS die Distanzen zu allen Knoten zu bestimmen, muss man noch ein zusätzliches Array führen

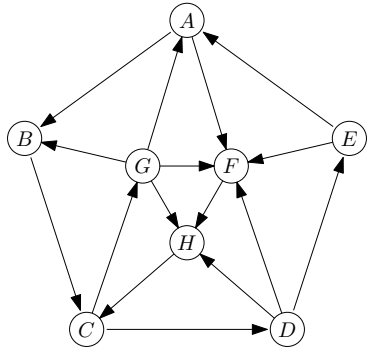
3. Feedback zu [code]expert

Allgemeines zu [code]expert

- Wird alles dieses Wochenende korrigiert
- Sorry für die Wartezeit

Fragen zu [code]expert eurerseits?

Tiefen- und Breitensuche



Start bei A

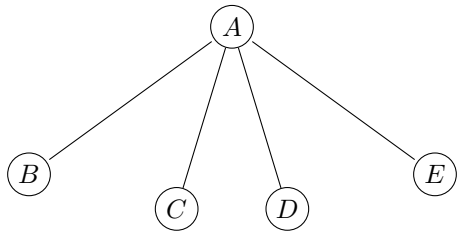
DFS: A, B, C, D, E, F, H, G

BFS: A, B, F, C, H, D, G, E

Es gibt keinen Startknoten, sodass die DFS-Ordnung der BFS-Ordnung entspricht.

Tiefen- und Breitensuche

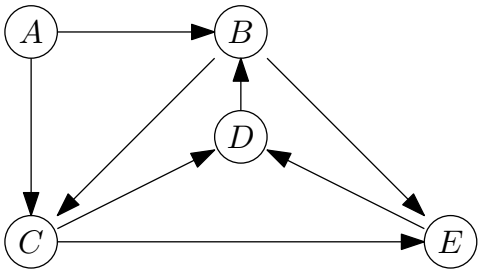
Stern: DFS-Ordnung entspricht BFS-Ordnung

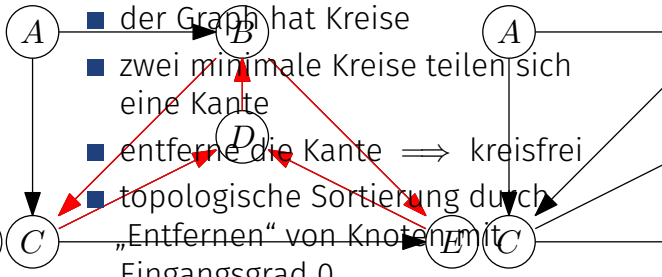


Start bei A
DFS: A, B, C, D, E
BFS: A, B, C, D, E

Start bei C
DFS: C, A, B, D, E
BFS: C, A, B, D, E

Topologische Sortierung



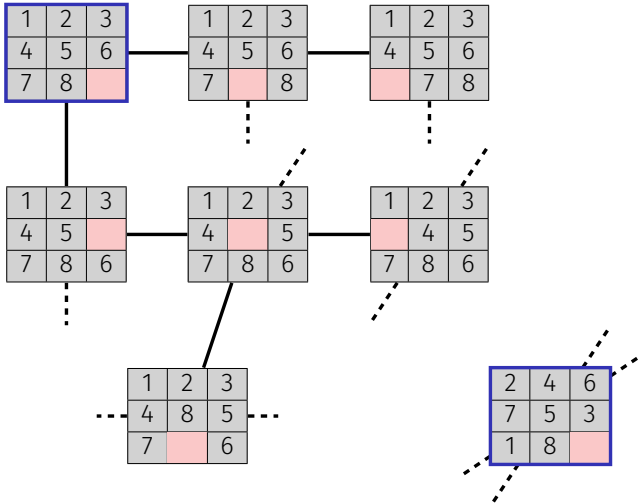


- der Graph hat Kreise
- zwei minimale Kreise teilen sich eine Kante
- entferne die Kante $\Longrightarrow$ kreisfrei
- topologische Sortierung durch „Entfernen“ von Knoten mit Eingangsgrad 0

Sliding Puzzle

Schnelleste Lösung finden für

Problem als Graph



Fragen/Unklarheiten?

5. All-pairs Shortest Path Problem

DP-Algorithmus Floyd-Warshall(G)

Input: Azyklischer Graph $G = (V, E, c)$
Output: Minimale Gewichte aller Pfade d

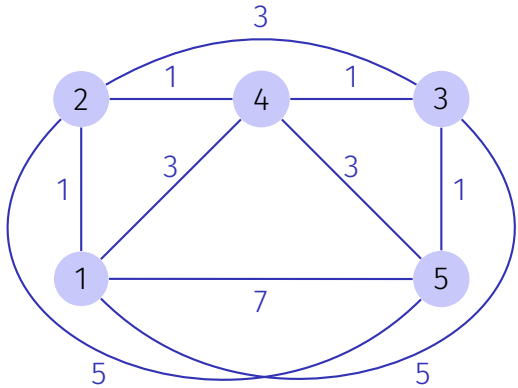
$d^0 \leftarrow c$
for $k \leftarrow 1$ **to** $|V|$ **do**
  for $i \leftarrow 1$ **to** $|V|$ **do**
    for $j \leftarrow 1$ **to** $|V|$ **do**
      $d^k(v_i, v_j) = \min\{d^{k-1}(v_i, v_j), d^{k-1}(v_i, v_k) + d^{k-1}(v_k, v_j)\}$

Laufzeit: $\Theta(|V|^3)$

Bemerkung: Der Algorithmus kann auf einer einzigen Matrix d (in place) ausgeführt werden.

▸ Videoempfehlung

Beispiel



Adjazenzmatrix $M = c$

	1	2	3	4	5
1	0	1	5	3	7
2	1	0	3	1	5
3	5	3	0	1	1
4	3	1	1	0	3
5	7	5	1	3	0

Beispiel

$k = 1$

0	1	5	3	7
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
7	5	1	3	0

d^0

$k = 2$

0	1	5	3	7
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
7	5	1	3	0

d^1

$k = 3$

0	1	4	2	6
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

$k = 4$

0	1	4	2	5
1	0	3	1	4
4	3	0	1	1
2	1	1	0	2
5	4	1	2	0

d^3

$k = 5$

0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

d^4

0	1	5	3	7
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
7	5	1	3	0

d^1

0	1	**4**	**2**	**6**
1	0	3	1	5
4	3	0	1	1
2	1	1	0	3
6	5	1	3	0

d^2

0	1	4	2	**5**
1	0	3	1	**4**
4	3	0	1	1
2	1	1	0	**2**
5	**4**	1	**2**	0

d^3

0	1	**3**	2	**4**
1	0	**2**	1	**3**
3	**2**	0	1	1
2	1	1	0	2
4	**3**	1	2	0

d^4

0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

d^5

Kürzester Weg für jedes Paar?

M

0	1	5	3	7
1	0	3	1	5
5	3	0	1	1
3	1	1	0	3
7	5	1	3	0

$D := d^5$

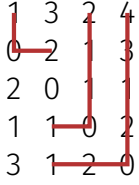
0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

D'

0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0

D''

0	1	3	2	4
1	0	2	1	3
3	2	0	1	1
2	1	1	0	2
4	3	1	2	0



Frage: Können wir die berechnete Matrix D verwenden, um den kürzesten Weg für jedes Knotenpaar zu bestimmen?

Direkte Wege klar: $i \to j$ wo $M[i,j] = D[i,j]$ (Siehe Markierungen in D' oben)

Könnten versuchen, den Algorithmus rückwärts laufen zu lassen. Beispiel $1 \to 3$ oben in D'': Kandidaten sind markiert. Finde mit absteigendem k den ersten passenden Kandidaten.

Kompliziert. Und uneffizient.

Idee

Merken uns jeweils das beste k für jedes (i, j) im Algorithmus in einer Matrix K.

Starten mit Matrix der existierenden Direktverbindungen (Kanten)

Beispiel

B

$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
0 1 5 3 7	0 1 **4 2 6**	0 1 4 2 **5**	0 1 **3** 2 **4**	0 1 3 2 4
1 0 3 1 5	1 0 3 1 5	1 0 3 1 **4**	1 0 **2** 1 **3**	1 0 2 1 3
5 3 0 1 1	**4** 3 0 1 1	4 3 0 1 1	**3 2** 0 1 1	3 2 0 1 1
3 1 1 0 3	**2** 1 1 0 3	2 1 1 0 **2**	2 1 1 0 2	2 1 1 0 2
7 5 1 3 0	**6** 5 1 3 0	**5 4** 1 **2** 0	**4 3** 1 2 0	4 3 1 2 0

K

1 2 3 4 5	1 2 **2 2 2**	1 2 2 2 **3**	1 2 **4** 2 **4**	1 2 4 2 4
1 2 3 4 5	1 2 3 4 5	1 2 3 4 **3**	1 2 **4 4 4**	1 2 4 4 4
1 2 3 4 5	**2** 2 3 4 5	2 2 3 4 5	**4 4** 3 4 5	4 4 3 4 5
1 2 3 4 5	**2** 2 3 4 5	2 2 3 4 **3**	2 2 3 4 3	2 2 3 4 3
1 2 3 4 5	**2** 2 3 4 5	**3 3** 3 **3** 5	**4 4** 3 3 5	4 4 3 3 5

Beispiel

K

	1	2	3	4	5
1	1	2	4	2	4
2	1	2	4	4	4
3	4	4	3	4	5
4	2	2	3	4	3
5	4	4	3	3	5

Wie liest man diese Matrix K?
Am Beispiel $1 \to 5$:

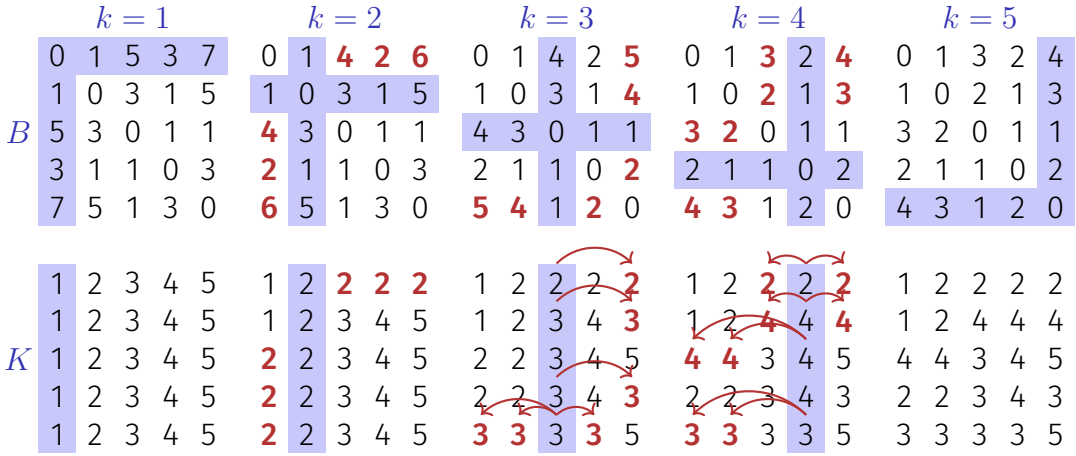
- Weg $1 \to 5$ geht über Knoten 4.
- Weg $1 \to 4$ geht über Knoten 2.
- Weg $4 \to 5$ geht über Knoten 3.
- Pfade $1 \to 2$ und $2 \to 4$ Sind direkt.
- Pfade $4 \to 3$ und $3 \to 5$ Sind direkt.

Insgesamt

$$1 \overset{4}{\to} 5 \quad \Rightarrow \quad 1 \overset{2}{\to} 4 \overset{3}{\to} 5 \quad \Rightarrow \quad 1 \to 2 \to 4 \to 3 \to 5$$

Rekonstruktion via Rekursion.
Alternative? Im Algorithmus Nachfolger merken

Beispiel



Vergleich der Verfahren

Algorithmus			Laufzeit
Dijkstra (Heap)	$c_v \geq 0$	1:n	$\mathcal{O}(\lvert E\rvert \log \lvert V\rvert)$
Dijkstra (Fibonacci-Heap)	$c_v \geq 0$	1:n	$\mathcal{O}(\lvert E\rvert + \lvert V\rvert \log \lvert V\rvert)$ *
Bellman-Ford		1:n	$\mathcal{O}(\lvert E\rvert \cdot \lvert V\rvert)$
Floyd-Warshall		n:n	$\Theta(\lvert V\rvert^3)$
Johnson		n:n	$\mathcal{O}(\lvert V\rvert \cdot \lvert E\rvert \cdot \log \lvert V\rvert)$
Johnson (Fibonacci-Heap)		n:n	$\mathcal{O}(\lvert V\rvert^2 \log \lvert V\rvert + \lvert V\rvert \cdot \lvert E\rvert)$ *

* amortisiert

Johnson (dieses Jahr nicht erklärt) ist besser als Floyd-Warshall nur für dünn besetzte Graphen ($|E| \approx \Theta(|V|)$).

Fragen/Unklarheiten?

6. Live [code]expert

Live [code]expert

’Lazy Deletion’ auf [code]expert

7. In-Class-Exercise (theoretisch)

In-Class-Exercises: Längster Pfad in DAGs

Das Kürzeste-Pfad-Problem hat einfache Lösungen (BFS, Dijkstra, Bellman-Ford). Das Längste-Pfad-Problem hingegen ist sehr schwierig! Für gerichtete Graphen gibt es vermutlich keinen schnellen Algorithmus, um Pfade der Länge $\gg \log^2 n$ zu finden.

Aufgabe:

Gegeben sei ein gerichteter, **kreisfreier** Graph (DAG) $G = (V, E)$.

Entwerfen Sie einen $\mathcal{O}(|V| + |E|)$-Laufzeit Algorithmus, um den *längsten Pfad* zu finden.

Tipp: G ist kreisfrei, Sie können also zuerst topologisch sortieren.

In-Class-Exercises: Längster Pfad in DAGs

Lösung:

1. Topologisch Sortieren. Laufzeit: $\mathcal{O}(|V| + |E|)$.
2. Berechne für jeden Knoten alle eingehenden Kanten: $\mathcal{O}(|V| + |E|)$.
3. Besuche jeden Knoten v in Reihenfolge der topologischen Sortierung und betrachte die Eingangs-Kanten: $\mathcal{O}(|V| + |E|)$.

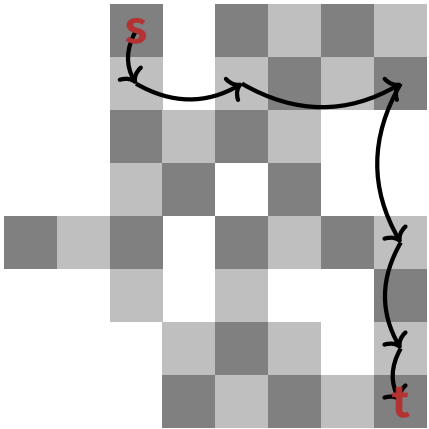
   $$\texttt{dist}[v] = \begin{cases} 0 & \text{keine Kanten,} \\ \max\limits_{(u,v)\in E} \{\texttt{dist}[u] + c(u,v)\} & \text{sonst.} \end{cases}$$

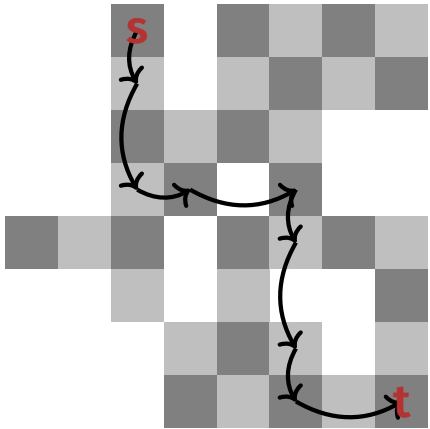
   Vorgänger merken!

Fragen/Unklarheiten?

8. Alte Prüfungsfragen (26.1.2018)

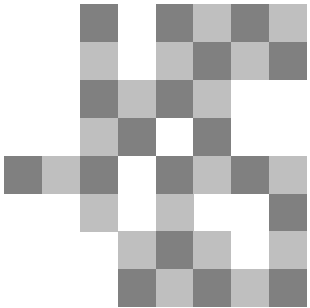
Kürzeste Wege Frage

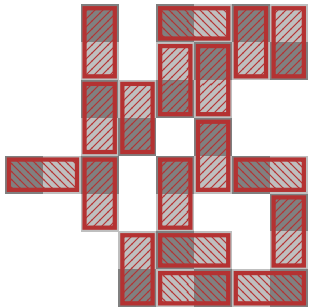




ist der dazugehörige Zustandsraum?

Max Flow Question





Wie bildet man das auf ein Max-Flow (Matching) Problem ab?

9. Outro

Allgemeine Fragen?

bei Fragen zu `[code]`$\mathrm{expert} \longrightarrow$ Mail ($+$ Geduld)

Bis zum nächsten Mal

Schönes Wochenende!