Algorithmen & Datenstrukturen

Übungsstunde 9

Michelle Honegger - CAB G59 - 17.11.2024

Miniquiz 8

- Passwort: **topologicalsorting7**
- 10 Fragen
- 12 Minuten
- 0.1 Punkte pro richtige Antwort

Programm

- Miniquiz
- Rückmeldung Serie 7
- Besprechung Aufgaben
- Theory Recap
- Prüfungsaufgaben
- Programmieraufgabe

Rückmeldung Serie 6

- 2-3-Trees nochmal anschauen, wenn nicht volle Punkte in 6.1
- Bei Invariantenbeweis auf Code eingehen
- Unterschied Momoization/DP beachten

Rückmeldung Serie 7

- Gut gelöst
- Bei DP müssen alle base cases angegeben werden

Demo 2-3-Trees

- https://n.ethz.ch/~mhonegger/2-3-Tree.html

Besprechung Miniquiz

Suppose that $G = (V, E)$ has an Eulerian Walk. Then we can always find an Eulerian Walk of G in time $O(|V|)$.

- Wahr
- Falsch

Let $G = (V, E)$ be *connected* and assume *all vertices of G have even degree*. Recall the algorithm $\mathrm{Euler\_Walk}$ from the lecture:

$\mathrm{Euler\_Walk}(u)$:

falls eine Kante $\{u, v\}$ existiert, die nicht markiert ist:

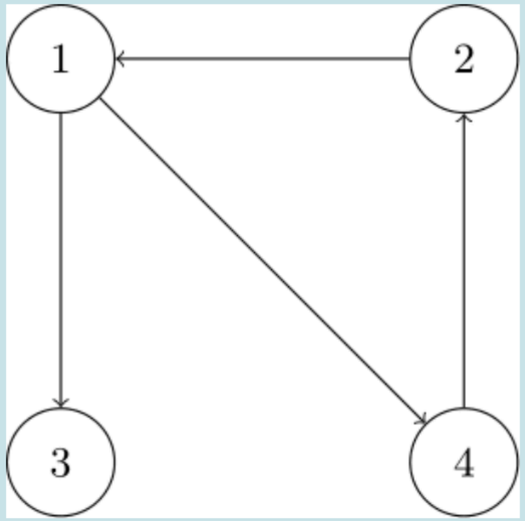
markiere die Kante $\{u, v\}$

$\mathrm{Euler\_Walk}(v)$

Suppose we start the recursion with a call to $\mathrm{Euler\_Walk}(u_0)$ for some vertex $u_0 \in V$.

True	False	
○	○	We have an even number of marked edges which contain u_0 immediately after the start of any call to $\mathrm{Euler\_Walk}(u_0)$.
○	○	Let $u' \in V$ with $u' \neq u_0$, then we have an even number of marked edges which contain u' immediately after the start of any call to $\mathrm{Euler\_Walk}(u')$.

Bewertung: **Teilpunkte** ?



What adjacency matrix describes the graph above? The indices are in increasing numerical order.

a. $\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

b. $\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

c. $\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$

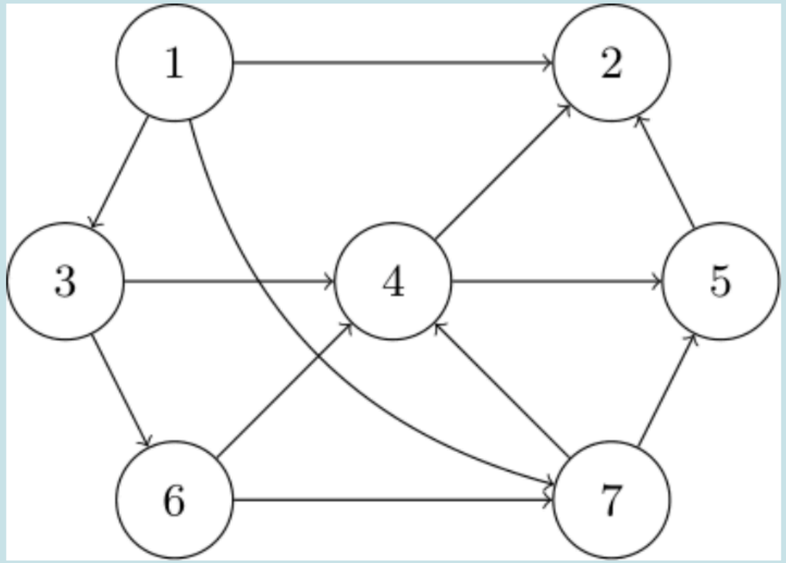
Suppose we restrict our inputs to graphs $G = (V, E)$ which are *connected* and satisfy $|E| \geq |V|^2/10$, rather than the class of all graphs. Then, depth-first search (DFS) runs (in the worst case over such graphs) asymptotically in the same time regardless of whether the graph is stored as adjacency lists or as an adjacency matrix.

- Wahr
- Falsch

Let G be a directed graph that contains a (directed) cycle. Which of the following statements are true for all such G?

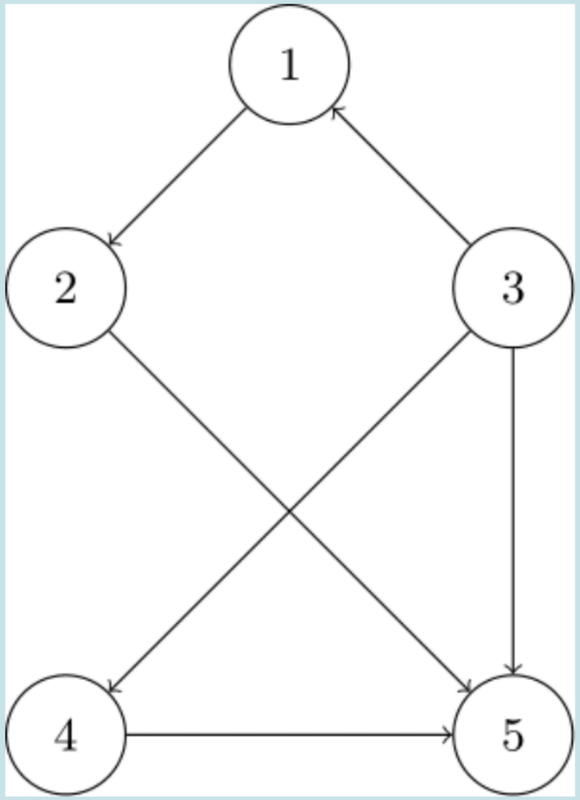
True	False	
		G has no sink vertex.
		G has no source vertex.
		G does not have a topological sorting of its vertices.

Bewertung: **Teilpunkte**



Which vertex is at the end of every topological sorting of the graph above?

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7



In every topological sorting of the graph above, vertex 1 must come before vertex 4.

- Wahr
- Falsch

We run *depth-first-search* (DFS) on a directed graph $G = (V, E)$ which contains a directed cycle. We determine all the *pre/post numbers*. What can we conclude about the relationship of pre/post numbers?

True	False	
○	○	For every edge $(u, v) \in E$, we have $\text{pre}(v) < \text{pre}(u) < \text{post}(u) < \text{post}(v)$.
○	○	There exists some edge $(u, v) \in E$ which has $\text{pre}(v) < \text{pre}(u) < \text{post}(u) < \text{post}(v)$.

Bewertung: **Teilpunkte** ?

We run *depth-first-search* (DFS) on a directed graph $G = (V, E)$ and we determine all the *pre/post numbers*. Then for an edge $(u, v) \in E$, it's possible that $\text{pre}(u) < \text{pre}(v) < \text{post}(u) < \text{post}(v)$.

- ○ Wahr
- ○ Falsch

Im obigen Graph zeigen die grünen, durchgezogenen Kanten einen *Tiefensuchbaum* an. Klassifizieren Sie die restlichen Kanten. (*Hinweis: Jede Option sollte **genau einmal** verwendet werden.*)

$(5, 4)$	Auswählen ...
$(1, 3)$	Auswählen ...
$(8, 1)$	Auswählen ...

Theory Recap

Gerichtete Graphen

- Kanten geordnete Paare: 

$e = (u,v) \in E$
- v ist **Nachfolger** von u und u ist **Vorgänger** von v
- Eingangsgrad (#Kanten, welche zu u zeigen) = $deg_{in}(u)$
- Ausgangsgrad (#Kanten, welche von u weggehen) = $deg_{out}(u)$
- **Quelle** wenn $deg_{in}(u)$=0
- **Senke** wenn $deg_{out}(u)$=0

Allgemein: $G(V,E)$

$|V| = n$

$|E| = m$

Graphen

Speicheroptionen

- Adjazenzmatrix: $A = (A_{uv})$

$$A_{uv} = \begin{cases} 1 & \text{, wenn } (u,v) \in E \\ 0 & \text{, sonst} \end{cases}$$

$O(n^2)$ Speicher

$O(n)$ um $\deg(u)$ zu finden

$O(1)$ um zu wissen ob es Kante von u nach v gibt.
um Kante zu löschen.
um Kante hinzuzufügen

Graphen

Speicheroptionen

- Adjazenzliste:

u	Adj(u)
1	All neighbors of 1
2	" 2
3	" 3
4	" 4

$O(n+m)$ Speicher

$O(\deg(u))$ um $\deg(u)$ zu finden

$O(\deg(u))$ um zu wissen ob es Kante von u nach v gibt.

$O(\deg(u))$ um Kante zu löschen.

um Kante hinzuzufügen

Topologische Sortierung

- Alle Abhängigkeiten erfüllt
- $\exists$ Topologische Sortierung $\Leftrightarrow \nexists$ gerichteter Zyklus
- $\Rightarrow$: Wenn jeder Knoten Nachfolger hat, ist keine Sortierung möglich
- $\Leftarrow$: Algorithmus
  - Finde Senke, entferne diese mache Rekursiv weiter
  - Wir suchen Langen Pfad von unmarkierten Knoten, letzter Knoten ist Senke

Topologische Sortierung

Algorithmen

```
Path(u)
  markiere u
  if ∃ Nachfolger v, unmarkiert
    Path (v)
```

```
Visit (u)
  markiere u
  for Nachfolger v, unmarkiert
      visit(v)
  füge v zu topol. sort. hinzu
```

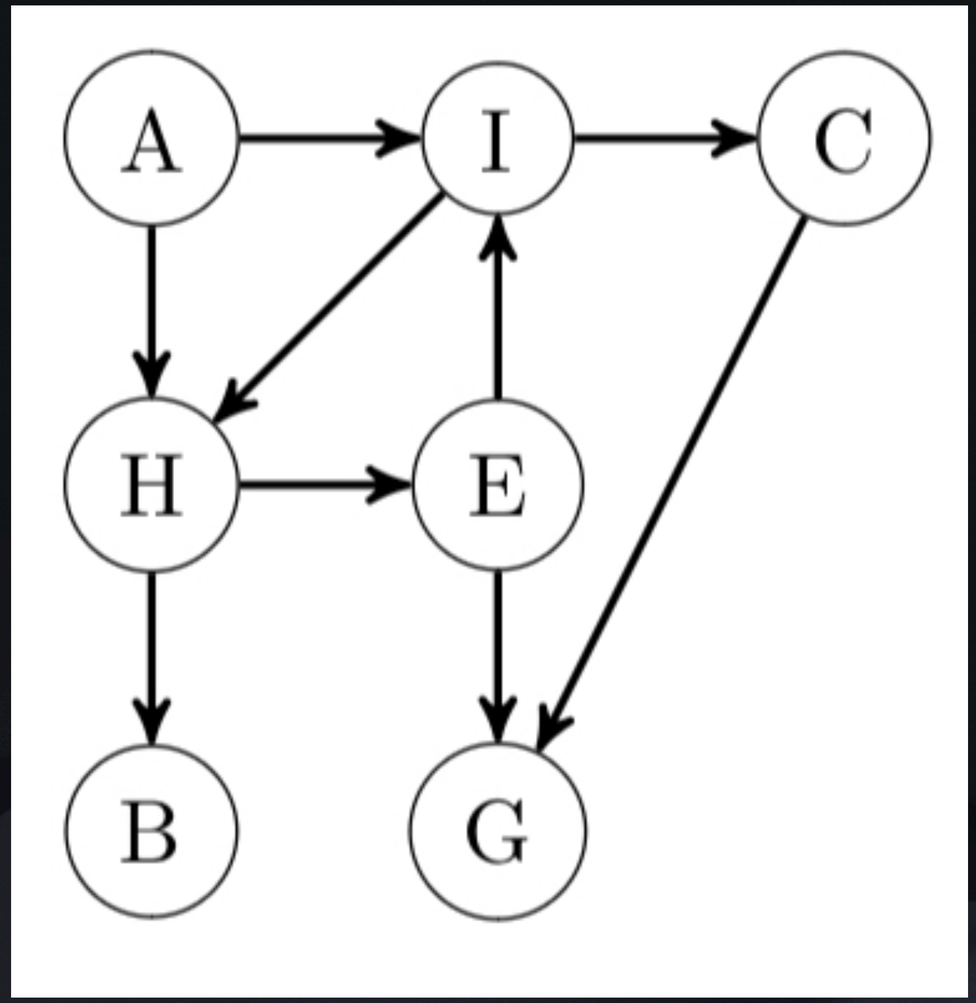
```
DFS(G)
  alle Knoten unmarkiert
  for u₀ ∈ V unmarkiert
      visit(u₀)
```

⇐

$O(n^2)$ mit Adjazenzmatrix

$O(n+m)$ mit Adjazenzliste

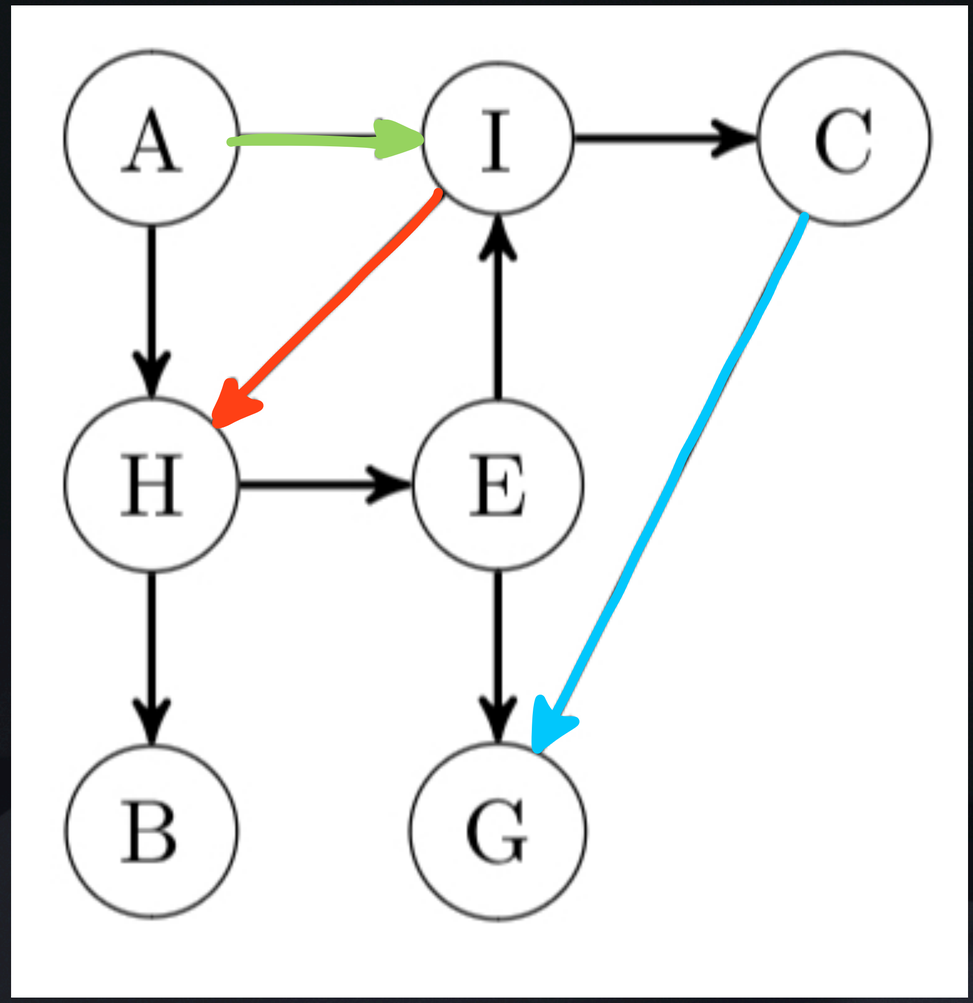
DFS - Prüfungsaufgabe



- Pre-Order:
- Post-Order:
- Intervalle:
  - Back Edges
  - Forward Edges
  - Cross Edges
- falls G Azyklisch, dann umgekehrte post-Order eine topologische Sortierung

DFS - Prüfungsaufgabe

- Pre-Order: A, H, B, E, G, I, C
- Post-Order: B, G, C, I, E, H, A
- Intervalle:
  - Back Edges
  - Forward Edges
  - Cross Edges



- falls G Azyklisch, dann umgekehrte post-Order eine topologische Sortierung

Exercise Recommendations

Aufgabenblatt 5

- 9.1 Eher kompliziert, gute Überlegungen und nützlich um sich an Notationen zu gewöhnen
- 9.2 Sehr wichtige Überlegungen, kommt oft als Prüfungsfrage
- 9.3 Bonus
- 9.4 Bonus

Peergrading

- Aufgabe8.3 d-f
- Grading Scheme in Lösungen beachten