Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet
  - sind benachbart

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet
  - sind benachbart
  - sind verbunden

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet
  - sind benachbart
  - sind verbunden
- gerichtete Graphen modellieren asymmetrische Beziehungen und Relationen wie

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet
  - sind benachbart
  - sind verbunden
- gerichtete Graphen modellieren asymmetrische Beziehungen und Relationen wie
  - ist abhängig von

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet
  - sind benachbart
  - sind verbunden
- gerichtete Graphen modellieren asymmetrische Beziehungen und Relationen wie
  - ist abhängig von
  - impliziert

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet
  - sind benachbart
  - sind verbunden
- gerichtete Graphen modellieren asymmetrische Beziehungen und Relationen wie
  - ist abhängig von
  - impliziert
  - muss vorher ausgeführt werden

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet
  - sind benachbart
  - sind verbunden
- gerichtete Graphen modellieren asymmetrische Beziehungen und Relationen wie
  - ist abhängig von
  - impliziert
  - muss vorher ausgeführt werden
  - ist besser als

Gerichtete Graphen

- ungerichtete Graphen können nur symmetrische Beziehungen und Relationen modellieren wie
  - sind befreundet
  - sind benachbart
  - sind verbunden
- gerichtete Graphen modellieren asymmetrische Beziehungen und Relationen wie
  - ist abhängig von
  - impliziert
  - muss vorher ausgeführt werden
  - ist besser als
- gerichtete Graphen werden insbesondere zur Modellierung von Abhängigkeiten und Abläufen genutzt

Gerichtete Graphen

Definition 1

Ein gericheter Graph oder Digraph (engl. directed graph) D ist ein Paar (V, A), wobei V eine endliche nicht-leere Menge von Knoten (engl. vertices) ist. Die Menge $A \subseteq V \times V$ ist eine Menge von gerichteten Kanten (engl. arcs).

Gerichtete Graphen

Definition 1

Ein gericheter Graph oder Digraph (engl. directed graph) D ist ein Paar (V, A), wobei V eine endliche nicht-leere Menge von Knoten (engl. vertices) ist. Die Menge $A \subseteq V \times V$ ist eine Menge von gerichteten Kanten (engl. arcs).

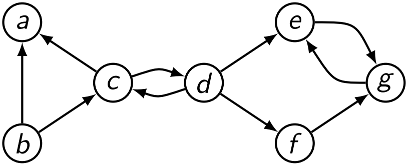
$$
\begin{aligned}
D &= (V, A) \\
V &= \{a, b, c, d, e, f, g\} \\
A &= \{(b, a), (b, c), (c, a), (c, d), \\
&\quad (d, c), (d, e), (d, f) \\
&\quad (e, g), (f, g), (g, e)\}
\end{aligned}
$$

Gerichtete Graphen

Definition 1

Ein gericheter Graph oder Digraph (engl. directed graph) D ist ein Paar (V, A), wobei V eine endliche nicht-leere Menge von Knoten (engl. vertices) ist. Die Menge $A \subseteq V \times V$ ist eine Menge von gerichteten Kanten (engl. arcs).

$$
\begin{aligned}
D &= (V, A) \\
V &= \{a, b, c, d, e, f, g\} \\
A &= \{(b, a), (b, c), (c, a), (c, d), \\
&\quad\; (d, c), (d, e), (d, f) \\
&\quad\; (e, g), (f, g), (g, e)\}
\end{aligned}
$$



Gerichtete Graphen

Definition 1

Ein gericheter Graph oder Digraph (engl. directed graph) D ist ein Paar (V, A), wobei V eine endliche nicht-leere Menge von Knoten (engl. vertices) ist. Die Menge $A \subseteq V \times V$ ist eine Menge von gerichteten Kanten (engl. arcs).

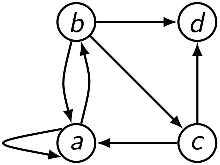
$$
\begin{aligned}
D &= (V, A) \\
V &= \{a, b, c, d\} \\
A &= \{(a, a), (a, b), (b, a), \\
&\qquad (c, a), (b, c), (b, d) \\
&\qquad (c, d)\}
\end{aligned}
$$

Gerichtete Graphen

Definition 1

Ein gericheter Graph oder Digraph (engl. directed graph) D ist ein Paar (V, A), wobei V eine endliche nicht-leere Menge von Knoten (engl. vertices) ist. Die Menge $A \subseteq V \times V$ ist eine Menge von gerichteten Kanten (engl. arcs).

$$
\begin{aligned}
D &= (V, A) \\
V &= \{a, b, c, d\} \\
A &= \{(a, a), (a, b), (b, a), \\
&\qquad (c, a), (b, c), (b, d) \\
&\qquad (c, d)\}
\end{aligned}
$$

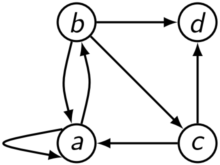


Gerichtete Graphen

Definition 1

Ein gericheter Graph oder Digraph (engl. directed graph) D ist ein Paar (V, A), wobei V eine endliche nicht-leere Menge von Knoten (engl. vertices) ist. Die Menge $A \subseteq V \times V$ ist eine Menge von gerichteten Kanten (engl. arcs).

$$
\begin{aligned}
D &= (V, A) \\
V &= \{a, b, c, d\} \\
A &= \{(a, a), (a, b), (b, a), \\
&\quad (c, a), (b, c), (b, d) \\
&\quad (c, d)\}
\end{aligned}
$$



- Kanten der Form (u, u) werden **Schleifen** genannt.

Eingangs- und Ausgangsgrade

Definition 2

Für einen Knoten v eines gerichteten Graphen $D = (V, A)$ definieren wir

$$\mathrm{indeg}(v) := \left|\{u \in V \mid (u, v) \in A\}\right|$$

als den Eingangsgrad von v.
Der Ausgangsgrad von v ist definiert als

$$\mathrm{outdeg}(v) := \left|\{u \in V \mid (v, u) \in A\}\right|$$

Eingangs- und Ausgangsgrade

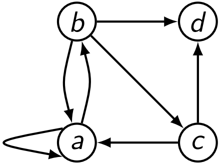
Definition 2

Für einen Knoten v eines gerichteten Graphen $D = (V, A)$ definieren wir

$$\text{indeg}(v) := \big|\{u \in V \mid (u, v) \in A\}\big|$$

als den Eingangsgrad von v.
Der Ausgangsgrad von v ist definiert als

$$\text{outdeg}(v) := \big|\{u \in V \mid (v, u) \in A\}\big|$$



Eingangs- und Ausgangsgrade

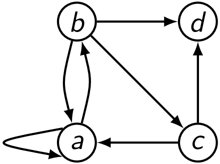
Definition 2

Für einen Knoten v eines gerichteten Graphen $D = (V, A)$ definieren wir

$$\mathrm{indeg}(v) := \left|\{u \in V \mid (u, v) \in A\}\right|$$

als den Eingangsgrad von v.
Der Ausgangsgrad von v ist definiert als

$$\mathrm{outdeg}(v) := \left|\{u \in V \mid (v, u) \in A\}\right|$$



- $\mathrm{indeg}(a) = 3$
- $\mathrm{outdeg}(a) = 2$

Eingangs- und Ausgangsgrade

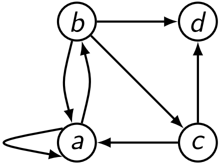
Definition 2

Für einen Knoten v eines gerichteten Graphen $D = (V, A)$ definieren wir

$$\text{indeg}(v) := \left|\{u \in V \mid (u, v) \in A\}\right|$$

als den Eingangsgrad von v.
Der Ausgangsgrad von v ist definiert als

$$\text{outdeg}(v) := \left|\{u \in V \mid (v, u) \in A\}\right|$$



- $\text{indeg}(a) = 3$
- $\text{outdeg}(a) = 2$
- $\text{indeg}(b) = 1$
- $\text{outdeg}(b) = 3$

Eingangs- und Ausgangsgrade

Definition 2

Für einen Knoten v eines gerichteten Graphen $D = (V, A)$ definieren wir

$$\text{indeg}(v) := \left|\{u \in V \mid (u, v) \in A\}\right|$$

als den Eingangsgrad von v.
Der Ausgangsgrad von v ist definiert als

$$\text{outdeg}(v) := \left|\{u \in V \mid (v, u) \in A\}\right|$$

Satz 3

Für jeden gerichteten Graphen $D = (V, A)$ gilt

$$\sum_{v \in V} \text{indeg}(v) = \sum_{v \in V} \text{outdeg}(v).$$

Eingangs- und Ausgangsgrade

Definition 2

Für einen Knoten v eines gerichteten Graphen $D = (V, A)$ definieren wir

$$\mathrm{indeg}(v) := \left|\{u \in V \mid (u, v) \in A\}\right|$$

als den Eingangsgrad von v.
Der Ausgangsgrad von v ist definiert als

$$\mathrm{outdeg}(v) := \left|\{u \in V \mid (v, u) \in A\}\right|$$

- Für einen Knoten $v \in V$ wird $\mathrm{indeg}(v) + \mathrm{outdeg}(v)$ auch der **Grad** von v genannt.

Wege in gerichteten Graphen

Definition 3

Ein Weg der Länge $l, l \in \mathbb{N}$, in einem gerichteten Graphen $D = (V, A)$ ist eine Folge $W = (v_0, v_1, \ldots, v_l)$ von Knoten aus V, so dass je zwei aufeinander folgende Knoten durch eine Kante miteinander verbunden sind, also

$$(v_i, v_{i+1}) \in A \quad \text{für alle } i = 0, \ldots, l-1.$$

v_0 wird Anfangsknoten und v_l wird Endknoten des Weges W genannt. Alle anderen Knoten werden innere Knoten genannt. Ein Pfad ist ein Weg, in dem alle Knoten paarweise verschieden sind.

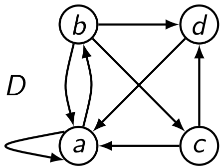
Wege in gerichteten Graphen

Definition 3

Ein Weg der Länge $l, l \in \mathbb{N}$, in einem gerichteten Graphen $D = (V, A)$ ist eine Folge $W = (v_0, v_1, \ldots, v_l)$ von Knoten aus V, so dass je zwei aufeinander folgende Knoten durch eine Kante miteinander verbunden sind, also

$$(v_i, v_{i+1}) \in A \quad \text{für alle } i = 0, \ldots, l-1.$$

v_0 wird Anfangsknoten und v_l wird Endknoten des Weges W genannt. Alle anderen Knoten werden innere Knoten genannt. Ein Pfad ist ein Weg, in dem alle Knoten paarweise verschieden sind.



- (b, d, a, a, b) ist ein Weg, aber kein Pfad in D.
- (c, a, b, d) ist ein Pfad in D.

Kreise in gerichteten Graphen

Definition 4

Ein Kreis der Länge $l, l \in \mathbb{N}$, in einem gerichteten Graphen $D = (V, A)$ ist eine Folge $C = (v_1, \ldots, v_l)$ von l Knoten, so dass

$$(v_l, v_1) \in A \quad \text{und} \quad (v_i, v_{i+1}) \in A \quad \text{für alle } i = 1, \ldots, l-1,$$

und diese l Kanten paarweise verschieden sind.

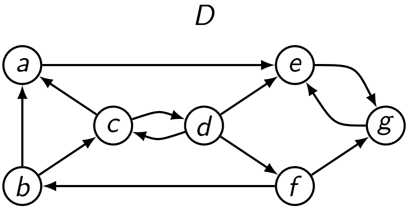
Kreise in gerichteten Graphen

Definition 4

Ein Kreis der Länge $l, l \in \mathbb{N}$, in einem gerichteten Graphen $D = (V, A)$ ist eine Folge $C = (v_1, \ldots, v_l)$ von l Knoten, so dass

$$(v_l, v_1) \in A \quad \text{und} \quad (v_i, v_{i+1}) \in A \quad \text{für alle } i = 1, \ldots, l-1,$$

und diese l Kanten paarweise verschieden sind.



- (b, c, d, f) ist ein Kreis in D.
- (c, d) ist ein Kreis in D.
- (a, e, d, c) ist **kein** Kreis in D.

Kreisfreie Graphen - DAGs

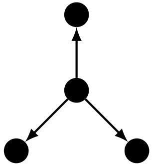
Definition 5

Ein gerichteter Graph $D = (V, A)$ heißt kreisfrei oder DAG (engl. directed acyclic graph), falls D keine Kreise besitzt.

Kreisfreie Graphen - DAGs

Definition 5

Ein gerichteter Graph $D = (V, A)$ heißt kreisfrei oder DAG (engl. directed acyclic graph), falls D keine Kreise besitzt.

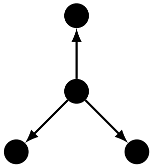


Graph ist ein DAG

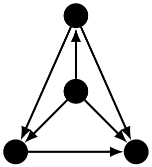
Kreisfreie Graphen - DAGs

Definition 5

Ein gerichteter Graph $D = (V, A)$ heißt kreisfrei oder DAG (engl. directed acyclic graph), falls D keine Kreise besitzt.



Graph ist ein DAG



Graph ist ein DAG

Projektorganisation I

- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.

Projektorganisation I

- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:

Projektorganisation I

- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.

Projektorganisation I

- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.
  2. Teilprojekt c muss abgeschlossen sein, bevor Teilprojekte b und e starten können.

Projektorganisation I

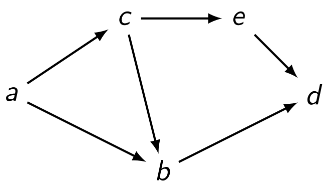
- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.
  2. Teilprojekt c muss abgeschlossen sein, bevor Teilprojekte b und e starten können.
  3. Teilprojekt d kann erst starten, nachdem Teilprojekte b und e abgeschlossen sind.

Projektorganisation I

- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.
  2. Teilprojekt c muss abgeschlossen sein, bevor Teilprojekte b und e starten können.
  3. Teilprojekt d kann erst starten, nachdem Teilprojekte b und e abgeschlossen sind.
- Kann das Projekt mit der Aufteilung in die Teilprojekte $a, \ldots, e$ bearbeitet werden?

Projektorganisation I

- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.
  2. Teilprojekt c muss abgeschlossen sein, bevor Teilprojekte b und e starten können.
  3. Teilprojekt d kann erst starten, nachdem Teilprojekte b und e abgeschlossen sind.
- Kann das Projekt mit der Aufteilung in die Teilprojekte $a, \ldots, e$ bearbeitet werden?



Graphische Darstellung der Abhängigkeiten von Teilprojekten

Topologische Sortierung

Definition 6

Eine topologische Sortierung eines gerichteten Graphen $D = (V, A)$ mit $|V| = n$ ist eine bijektive Funktion $s : V \to \{1, \ldots, n\}$, so dass

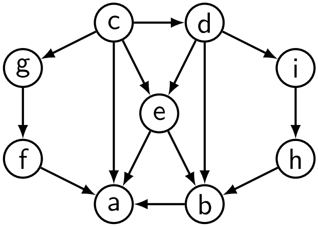
$$s(u) < s(v) \quad \text{für alle Kanten } (u, v) \in A.$$

Topologische Sortierung

Definition 6

Eine topologische Sortierung eines gerichteten Graphen $D = (V, A)$ mit $|V| = n$ ist eine bijektive Funktion $s : V \to \{1, \ldots, n\}$, so dass

$$s(u) < s(v) \quad \text{für alle Kanten } (u, v) \in A.$$



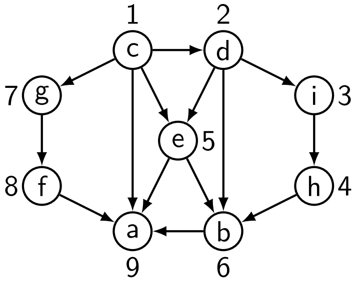
gerichteter Graph

Topologische Sortierung

Definition 6

Eine topologische Sortierung eines gerichteten Graphen $D = (V, A)$ mit $|V| = n$ ist eine bijektive Funktion $s : V \to \{1, \ldots, n\}$, so dass

$$s(u) < s(v) \quad \text{für alle Kanten } (u, v) \in A.$$



gerichteter Graph mit topologischer Sortierung

Topologische Sortierung

Definition 6

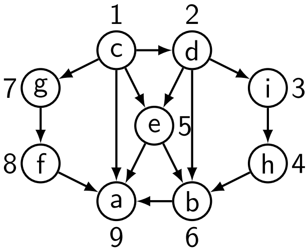
Eine topologische Sortierung eines gerichteten Graphen $D = (V, A)$ mit $|V| = n$ ist eine bijektive Funktion $s : V \to \{1, \ldots, n\}$, so dass

$$s(u) < s(v) \quad \text{für alle Kanten } (u, v) \in A.$$

Satz 7

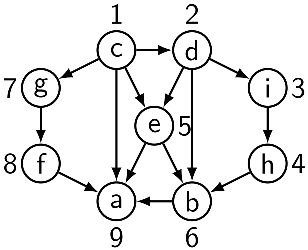
Ein gerichteter Graph $D = (V, A)$ besitzt genau dann eine topologische Sortierung, wenn er kreisfrei ist.
Also, DAGs und nur DAGs besitzen eine topologische Sortierung.

Topologische Sortierung

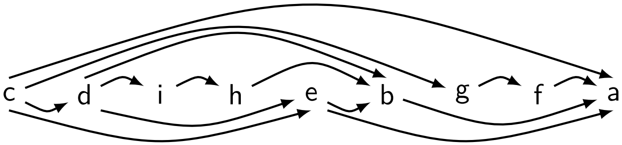


gerichteter Graph mit topologischer Sortierung

Topologische Sortierung



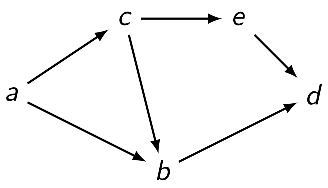
gerichteter Graph mit topologischer Sortierung



Toplogische Sortierung als lineare Anordnung der Knoten

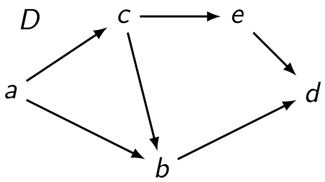
Projektorganisation I

- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.
  2. Teilprojekt c muss abgeschlossen sein, bevor Teilprojekte b und e starten können.
  3. Teilprojekt d kann erst starten, nachdem Teilprojekte b und e abgeschlossen sind.
- Kann das Projekt mit der Aufteilung in die Teilprojekte $a, \ldots, e$ bearbeitet werden?



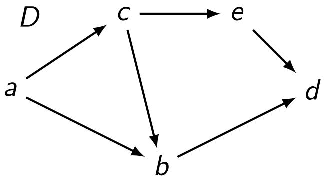
Graphische Darstellung der Abhängigkeiten von Teilprojekten

Projektorganisation I



- D ist ein DAG.

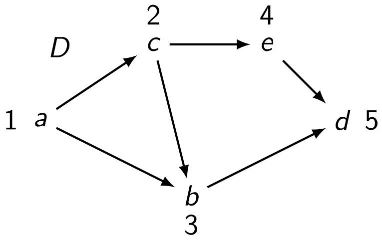
Projektorganisation I



- D ist ein DAG.

$\Rightarrow$ D besitzt eine topologische Sortierung.

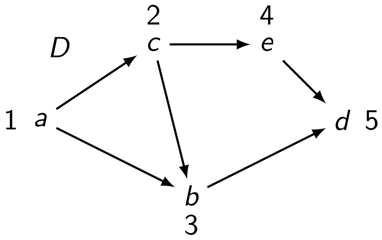
Projektorganisation I



- D ist ein DAG.

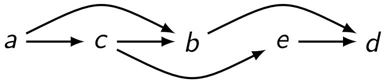
$\Rightarrow$ D besitzt eine topologische Sortierung.

Projektorganisation I



- D ist ein DAG.

$\Rightarrow$ D besitzt eine topologische Sortierung.



Mögliche Reihenfolge der Teilprojekte

Projektorganisation II

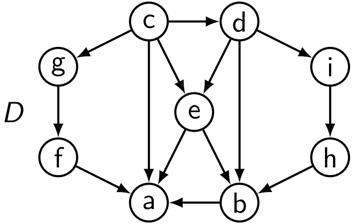
- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.
  2. Teilprojekt c muss abgeschlossen sein, bevor Teilprojekte b und e starten können.
  3. Teilprojekt d kann erst starten, nachdem Teilprojekte b und e abgeschlossen sind.

Projektorganisation II

- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.
  2. Teilprojekt c muss abgeschlossen sein, bevor Teilprojekte b und e starten können.
  3. Teilprojekt d kann erst starten, nachdem Teilprojekte b und e abgeschlossen sind.
- Jedes der Teilprojekte benötigt zwei Stunden zu seiner Bearbeitung.
- Um die Bearbeitung des Projekts zu beschleunigen, werden mehrere Teams gebildet. Jedes der Teams kann jedes der Teilprojekte bearbeiten.
- Wie viele Teams sollten gebildet werden? Wie lange dauert die Bearbeitung des gesamten Projekts?

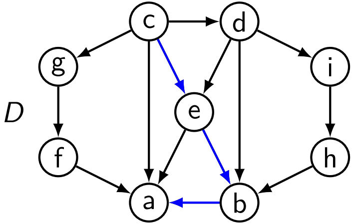
Kritische Pfade in gerichteten Graphen

- Sei $D = (V, A)$ ein gerichteter, kreisfeier Graph und $W = (v_0, \ldots, v_l)$ ein Weg in D.
- l heißt die **Länge** des Wegs W.
- Die Länge eines Wegs ist also die Anzahl der Kanten des Wegs.
- Ein Pfad maximaler Länge in D heißt **kritischer Pfad** von D.



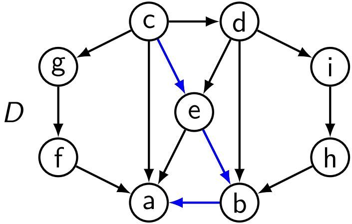
Kritische Pfade in gerichteten Graphen

- Sei $D = (V, A)$ ein gerichteter, kreisfeier Graph und $W = (v_0, \ldots, v_l)$ ein Weg in D.
- l heißt die **Länge** des Wegs W.
- Die Länge eines Wegs ist also die Anzahl der Kanten des Wegs.
- Ein Pfad maximaler Länge in D heißt **kritischer Pfad** von D.



Kritische Pfade in gerichteten Graphen

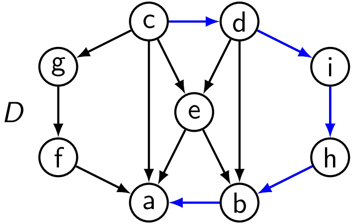
- Sei $D = (V, A)$ ein gerichteter, kreisfeier Graph und $W = (v_0, \ldots, v_l)$ ein Weg in D.
- l heißt die **Länge** des Wegs W.
- Die Länge eines Wegs ist also die Anzahl der Kanten des Wegs.
- Ein Pfad maximaler Länge in D heißt **kritischer Pfad** von D.



- Weg (sogar Pfad) der Länge 3 in D

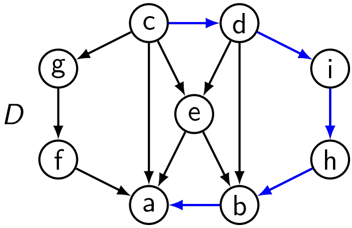
Kritische Pfade in gerichteten Graphen

- Sei $D = (V, A)$ ein gerichteter, kreisfeier Graph und $W = (v_0, \dots, v_l)$ ein Weg in D.
- l heißt die **Länge** des Wegs W.
- Die Länge eines Wegs ist also die Anzahl der Kanten des Wegs.
- Ein Pfad maximaler Länge in D heißt **kritischer Pfad** von D.



Kritische Pfade in gerichteten Graphen

- Sei $D = (V, A)$ ein gerichteter, kreisfeier Graph und $W = (v_0, \ldots, v_l)$ ein Weg in D.
- l heißt die **Länge** des Wegs W.
- Die Länge eines Wegs ist also die Anzahl der Kanten des Wegs.
- Ein Pfad maximaler Länge in D heißt **kritischer Pfad** von D.

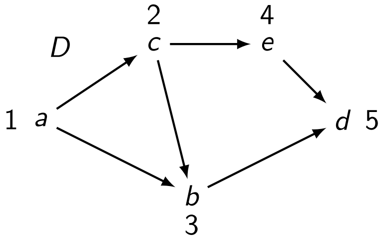


- Pfad der Länge 5 in D
- ein kritischer Pfad in D

Projektorganisation II

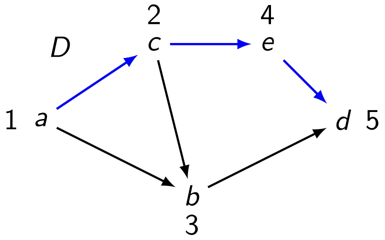
- Ein großes Projekt wird in fünf Teilprojekte a, b, c, d, e aufgeteilt, um die Bearbeitung zu vereinfachen und zu beschleunigen.
- Dabei existieren allerdings folgende Abhängigkeiten zwischen den einzelnen Teilprojekten:
  1. Teilprojekte b und c benötigen die Ergebnisse von Teilprojekt a.
  2. Teilprojekt c muss abgeschlossen sein, bevor Teilprojekte b und e starten können.
  3. Teilprojekt d kann erst starten, nachdem Teilprojekte b und e abgeschlossen sind.
- Jedes der Teilprojekte benötigt zwei Stunden zu seiner Bearbeitung.
- Um die Bearbeitung des Projekts zu beschleunigen, werden mehrere Teams gebildet. Jedes der Teams kann jedes der Teilprojekte bearbeiten.
- Wie viele Teams sollten gebildet werden? Wie lange dauert die Bearbeitung des gesamten Projekts?

Projektorganisation II



- gerichteter Graph der Abhängigkeiten der Teilprojekte
- topologische Sortierung

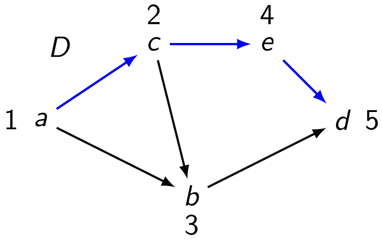
Projektorganisation II



- gerichteter Graph der Abhängigkeiten der Teilprojekte
- topologische Sortierung

- ein kritischer Pfad der Länge 3

Projektorganisation II

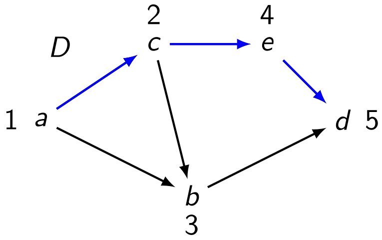


- gerichteter Graph der Abhängigkeiten der Teilprojekte
- topologische Sortierung

- ein kritischer Pfad der Länge 3

$\Rightarrow$ die Ausführung des Projekts benötigt mindestens 8 Stunden

Projektorganisation II



- gerichteter Graph der Abhängigkeiten der Teilprojekte
- topologische Sortierung

- ein kritischer Pfad der Länge 3

$\Rightarrow$ die Ausführung des Projekts benötigt mindestens 8 Stunden

$\Rightarrow$ zwei Teams genügen, um das Projekt in 8 Stunden zu beenden

Gerichtete und ungerichtete Graphen

Zugrunde liegender Graph

Jedem gerichteten Graphen $D = (V, A)$ kann ein ungerichteter Graph $G = (V, E)$ zugeordnet werden, indem

1. jede gerichtete Kante $(u, v), u \neq v$, durch die ungerichtete Kante $\{u, v\}$ ersetzt wird
2. Schleifen entfernt werden
3. danach mehrfach auftretende Kante durch eine Kante ersetzt werden.

Der so entstehende ungerichtete Grapph G heißt der D zugrunde liegende ungerichtete Graph.

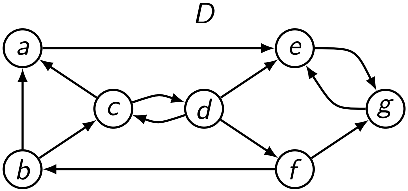
Gerichtete und ungerichtete Graphen

Zugrunde liegender Graph

Jedem gerichteten Graphen $D = (V, A)$ kann ein ungerichteter Graph $G = (V, E)$ zugeordnet werden, indem

1. jede gerichtete Kante $(u, v), u \neq v$, durch die ungerichtete Kante $\{u, v\}$ ersetzt wird
2. Schleifen entfernt werden
3. danach mehrfach auftretende Kante durch eine Kante ersetzt werden.

Der so entstehende ungerichtete Grapph G heißt der D zugrunde liegende ungerichtete Graph.



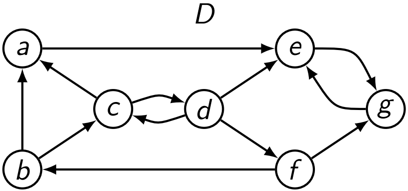
Gerichtete und ungerichtete Graphen

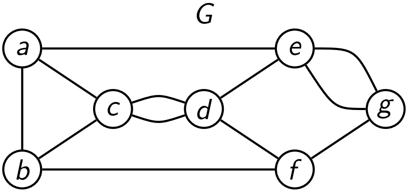
Zugrunde liegender Graph

Jedem gerichteten Graphen $D = (V, A)$ kann ein ungerichteter Graph $G = (V, E)$ zugeordnet werden, indem

1. jede gerichtete Kante $(u, v), u \neq v$, durch die ungerichtete Kante $\{u, v\}$ ersetzt wird
2. Schleifen entfernt werden
3. danach mehrfach auftretende Kante durch eine Kante ersetzt werden.

Der so entstehende ungerichtete Grapph G heißt der D zugrunde liegende ungerichtete Graph.





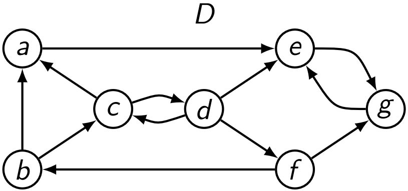
Gerichtete und ungerichtete Graphen

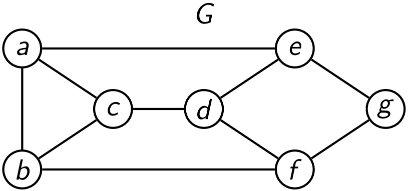
Zugrunde liegender Graph

Jedem gerichteten Graphen $D = (V, A)$ kann ein ungerichteter Graph $G = (V, E)$ zugeordnet werden, indem

1. jede gerichtete Kante $(u, v), u \neq v$, durch die ungerichtete Kante $\{u, v\}$ ersetzt wird
2. Schleifen entfernt werden
3. danach mehrfach auftretende Kante durch eine Kante ersetzt werden.

Der so entstehende ungerichtete Grapph G heißt der D zugrunde liegende ungerichtete Graph.





Zusammenhängende gerichtete Graphen

Ein (gerichteter) Pfad im gerichteten Graphen $D = (V, A)$ mit Anfangsknoten u und Endknoten v heißt gerichteter u-v-**Pfad** in D.

Zusammenhängende gerichtete Graphen

Ein (gerichteter) Pfad im gerichteten Graphen $D = (V, A)$ mit Anfangsknoten u und Endknoten v heißt gerichteter u-v-**Pfad** in D.

Definition 8

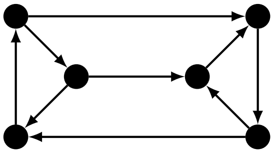
Ein gerichteter Graph $D = (V, A)$ heißt stark zusammenhängend, wenn für jedes Paar von Knoten $u, v \in V, u \neq v$, ein gerichteter u-v-Pfad in D existiert.
Ein gerichteter Graph $D = (V, A)$ heißt schwach zusammenhängend, wenn der zugrunde liegende ungerichtete Graph $G = (V, E)$ zusammenhängend ist.

Zusammenhängende gerichtete Graphen

Ein (gerichteter) Pfad im gerichteten Graphen $D = (V, A)$ mit Anfangsknoten u und Endknoten v heißt gerichteter u-v-**Pfad** in D.

Definition 8

Ein gerichteter Graph $D = (V, A)$ heißt stark zusammenhängend, wenn für jedes Paar von Knoten $u, v \in V, u \neq v$, ein gerichteter u-v-Pfad in D existiert.
Ein gerichteter Graph $D = (V, A)$ heißt schwach zusammenhängend, wenn der zugrunde liegende ungerichtete Graph $G = (V, E)$ zusammenhängend ist.



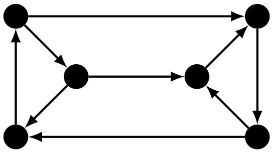
stark zusammenhängend

Zusammenhängende gerichtete Graphen

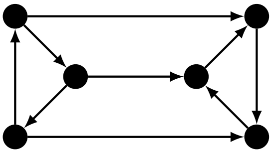
Ein (gerichteter) Pfad im gerichteten Graphen $D = (V, A)$ mit Anfangsknoten u und Endknoten v heißt gerichteter u-v-**Pfad** in D.

Definition 8

Ein gerichteter Graph $D = (V, A)$ heißt stark zusammenhängend, wenn für jedes Paar von Knoten $u, v \in V, u \neq v$, ein gerichteter u-v-Pfad in D existiert.
Ein gerichteter Graph $D = (V, A)$ heißt schwach zusammenhängend, wenn der zugrunde liegende ungerichtete Graph $G = (V, E)$ zusammenhängend ist.



stark zusammenhängend



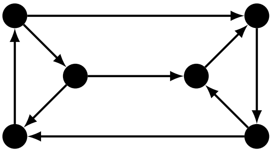
schwach zusammenhängend

Zusammenhängende gerichtete Graphen

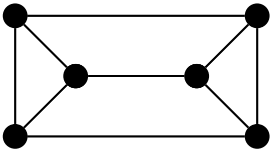
Ein (gerichteter) Pfad im gerichteten Graphen $D = (V, A)$ mit Anfangsknoten u und Endknoten v heißt gerichteter u-v-**Pfad** in D.

Definition 8

Ein gerichteter Graph $D = (V, A)$ heißt stark zusammenhängend, wenn für jedes Paar von Knoten $u, v \in V, u \neq v$, ein gerichteter u-v-Pfad in D existiert.
Ein gerichteter Graph $D = (V, A)$ heißt schwach zusammenhängend, wenn der zugrunde liegende ungerichtete Graph $G = (V, E)$ zusammenhängend ist.



stark zusammenhängend



Teilgraphen und induzierte Teilgraphen

Definition 9

Ein gerichteter Graph $H = (V_H, A_H)$ heißt (schwacher) Teilgraph eines gerichteten Graphen $D = (V_D, A_D)$, falls

$$V_H \subseteq V_D \quad \text{und} \quad A_H \subseteq A_D.$$

Enthält A_H alle Kanten aus A_D, deren inzidente Knoten in V_H liegen, also

$$A_H = A_D \cap (V_H \times V_H),$$

so nennt man H einen induzierten Teilgraphen von D.

Teilgraphen und induzierte Teilgraphen

Definition 9

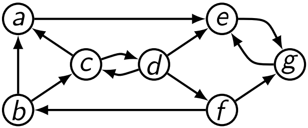
Ein gerichteter Graph $H = (V_H, A_H)$ heißt (schwacher) Teilgraph eines gerichteten Graphen $D = (V_D, A_D)$, falls

$$V_H \subseteq V_D \quad \text{und} \quad A_H \subseteq A_D.$$

Enthält A_H alle Kanten aus A_D, deren inzidente Knoten in V_H liegen, also

$$A_H = A_D \cap (V_H \times V_H)\,,$$

so nennt man H einen induzierten Teilgraphen von D.



Teilgraphen und induzierte Teilgraphen

Definition 9

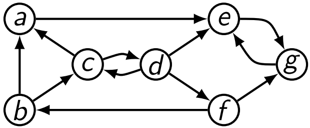
Ein gerichteter Graph $H = (V_H, A_H)$ heißt (schwacher) Teilgraph eines gerichteten Graphen $D = (V_D, A_D)$, falls

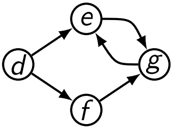
$$V_H \subseteq V_D \quad \text{und} \quad A_H \subseteq A_D.$$

Enthält A_H alle Kanten aus A_D, deren inzidente Knoten in V_H liegen, also

$$A_H = A_D \cap (V_H \times V_H),$$

so nennt man H einen induzierten Teilgraphen von D.





Durch d, e, f, g induzierter Teilgraph

Programmablaufgraphen

- modellieren Abläufe durch ein verzweigtes Programm
- eingesetzt in Compilern und Analysewerkzeugen der Softwareentwicklung

Programmablaufgraphen

```
ug = 0;                              // A
og = obereGrenze;
while ug <= og do                    // B
    mitte = (ug + og)/2;             // C
    if a[mitte] == x then
        return mitte;                // H
    else if a[mitte] < x then        // D
        ug = mitte + 1;              // E
    else                             // F
        og = mitte − 1
    end
end
return nichtGefunden;                // G
```

- modellieren Abläufe durch ein verzweigtes Programm
- eingesetzt in Compilern und Analysewerkzeugen der Softwareentwicklung

Programmablaufgraphen

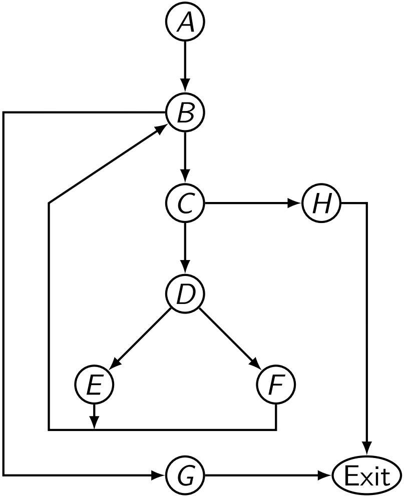
```
ug = 0;                                // A
og = obereGrenze;
while ug <= og do                      // B
 |  mitte = (ug + og)/2;               // C
 |  if a[mitte] == x then
 |   |  return mitte;                  // H
 |  else if a[mitte] < x then          // D
 |   |  ug = mitte + 1;                // E
 |  else                               // F
 |   |  og = mitte − 1
 |  end
end
return nichtGefunden;                  // G
```

- modellieren Abläufe durch ein verzweigtes Programm
- eingesetzt in Compilern und Analysewerkzeugen der Softwareentwicklung

- Knoten entsprechen unverzweigten Anweisungsfolgen mit Verzweigung am Ende
- Kanten führen zu möglichen Nachfolgern im Programmablauf

Programmablaufgraphen

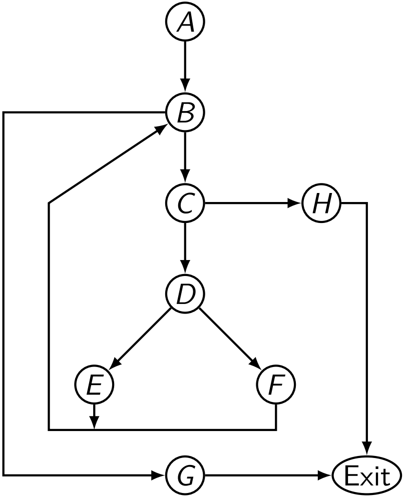
```
ug = 0;                                  // A
og = obereGrenze;
while ug <= og do                        // B
    mitte = (ug + og)/2;                 // C
    if a[mitte] == x then
        return mitte;                    // H
    else if a[mitte] < x then            // D
        ug = mitte + 1;                  // E
    else                                 // F
        og = mitte − 1
    end
end
return nichtGefunden;                    // G
```



Programmablaufgraphen

Fragen und Aufgaben

- Menge von Wegen, die alle Kanten überdecken (Testen der Software)
- Wege mit bestimmten Eigenschaften (Datenflussanalyse)

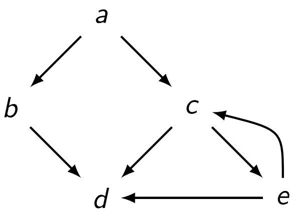


Aufrufgraphen

- modellieren Aufrufbeziehungen zwischen Funktionen in einem Programm
- eingesetzt in Compilern und Analysewerkzeugen der Softwareentwicklung

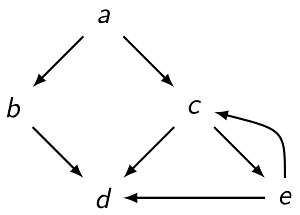
Aufrufgraphen

- modellieren Aufrufbeziehungen zwischen Funktionen in einem Programm
- eingesetzt in Compilern und Analysewerkzeugen der Softwareentwicklung



Aufrufgraphen

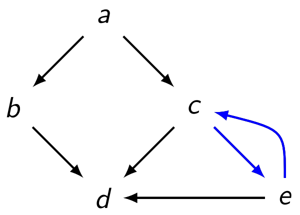
- modellieren Aufrufbeziehungen zwischen Funktionen in einem Programm
- eingesetzt in Compilern und Analysewerkzeugen der Softwareentwicklung



- Knoten entsprechen Aufrufen im Programm
- Kante (a, b) bedeutet: Funktion a könnte Funktion b aufrufen

Aufrufgraphen

- modellieren Aufrufbeziehungen zwischen Funktionen in einem Programm
- eingesetzt in Compilern und Analysewerkzeugen der Softwareentwicklung



- Knoten entsprechen Aufrufen im Programm
- Kante (a, b) bedeutet: Funktion a könnte Funktion b aufrufen

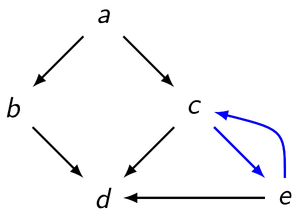
- **Kreise** entsprechen Funktionen, die sich wechselweise rekursiv aufrufen.
- Beispiel (c, e): c ruft e auf, die wiederum c aufruft.

Aufrufgraphen

Fragen und Aufgaben

- Welche Funktionen sind rekursiv?
- Welche Funktionen sind nicht (mehr) erreichbar?
- Indirekte Wirkung von Aufrufen:
  - nur e verändere globale Variable x

  $\Rightarrow$ Aufrufe von b und d lassen x unverändert



- **Kreise** entsprechen Funktionen, die sich wechselweise rekursiv aufrufen.
- Beispiel (c, e): c ruft e auf, die wiederum c aufruft.