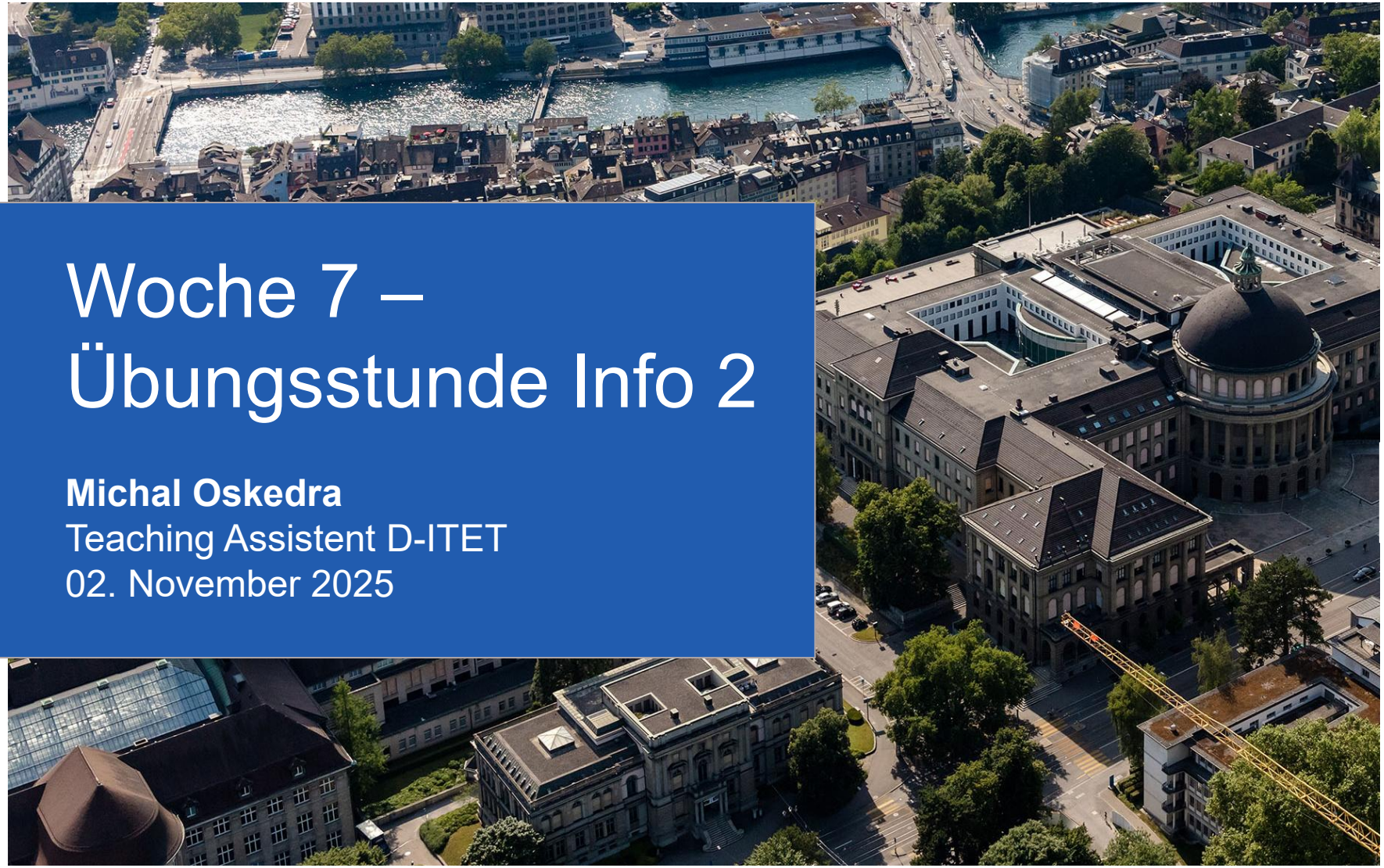
ETH zürich

Info2 - ML J 34.1 14-16

WhatsApp group



Woche 7 – Übungsstunde Info 2

Michal Oskedra
Teaching Assistent D-ITET
02. November 2025

Webseite:
moskedra-web.ethz.ch

01 – Theorie

01.1 Graphen Definitionen

01.2 Tiefensuche

01.3 Breitensuche

01.4 Topologisches Sortieren

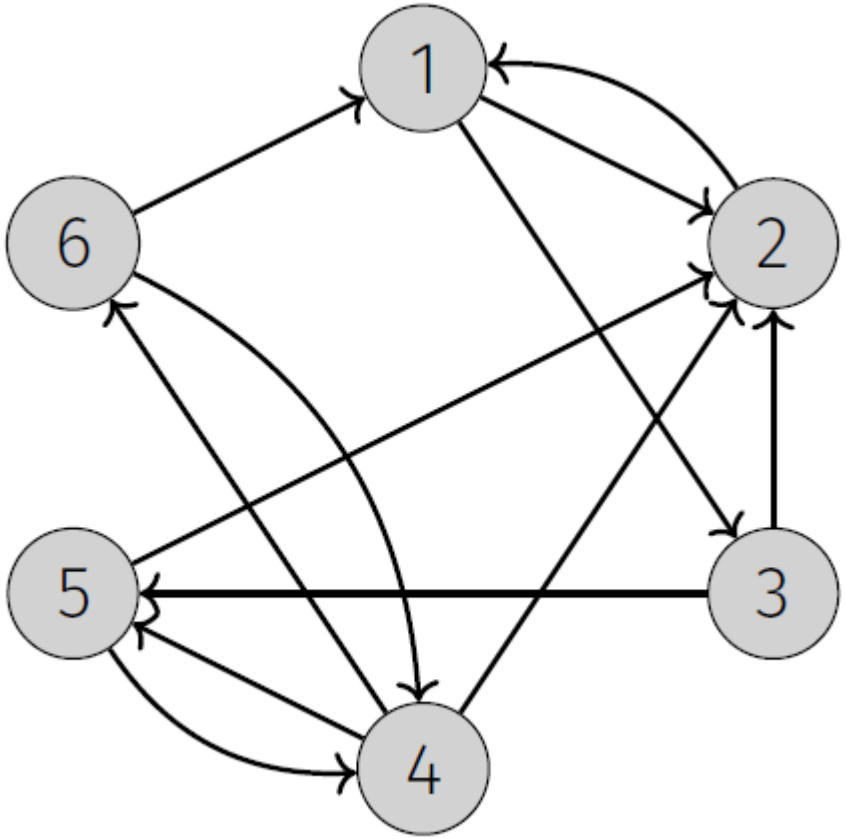
Graph

Graph $G = (V, E)$

Knoten (nodes/vertices) V

Kanten (edges) $E \subseteq V \times V$

Ungerichtet, falls $V_i \rightarrow V_j \Rightarrow V_j \rightarrow V_i$

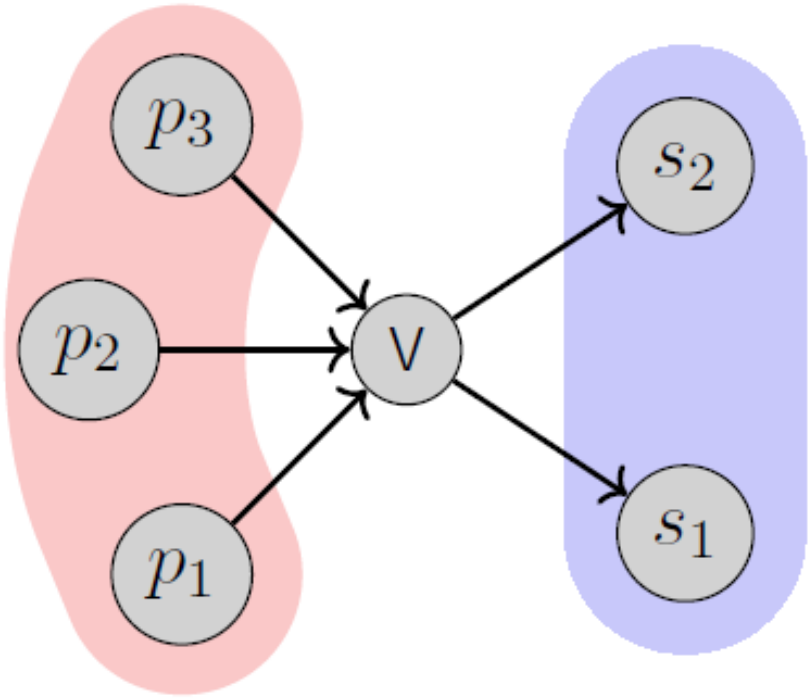


$$V = \{1, 2, 3, 4, 5, 6\}$$

$$E = \{(3,5), (1,3), (4,6), (4,5), (5,4), (2,1), (1,2), (3,2), (6,1), (6,4), (4,2), (5,2)\}$$

Adjazente Knoten

adjazent = benachbart



$N^-(v)$

Vorgänger

$N^+(v)$

Nachfolger

Grad einer Knote

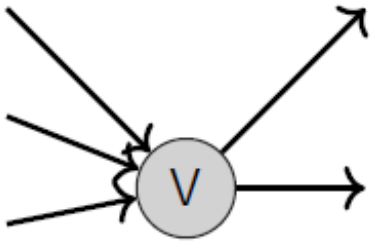
Eingangsgrad (in-degree) =
Vorgänger einer Kante

$$\deg^-(v) = |N^-(v)|$$

Ausgangsgrad (out-degree)
= # Nachfolger einer Kante

$$\deg^+(v) = |N^+(v)|$$

Beispiel:



$$\deg^-(v) = 3, \deg^+(v) = 2$$



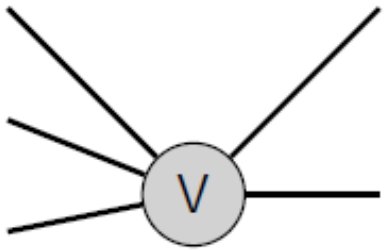
$$\deg^-(w) = 1, \deg^+(w) = 1$$

Ungerichtete Graphen

Grad = Anzahl von Kanten,
die an Knoten anliegen

$$\deg(v) = |N(v)|$$

Beispiel:



$\deg(v) = 5$

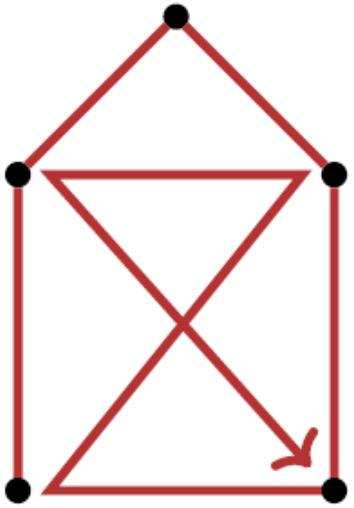


$\deg(w) = 2$

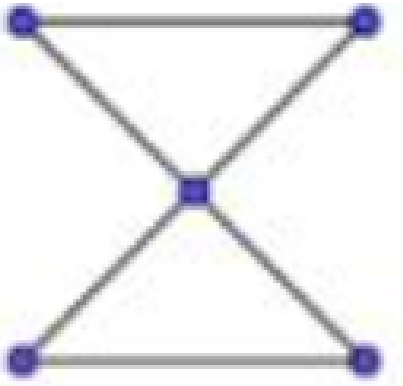
Runden um den Graph

Angewendet bei ungerichteten Graphen

Eulersweg (Eulerian path):
Ein **Weg**, der jede Kante genau einmal benutzt (Endet **nicht** an gleichen Knoten)

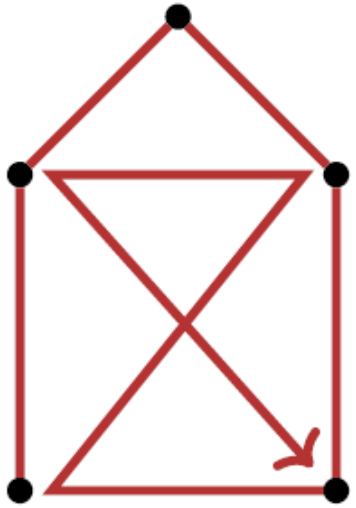


Eulerkreis (Eulerian cycle):
Ein **Kreis**, der jede Kante genau einmal benutzt
(Endet an gleichen Knoten)

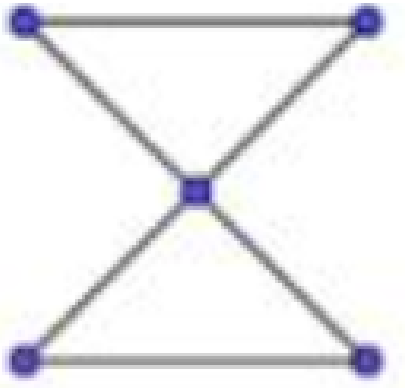


Runden um den Graph

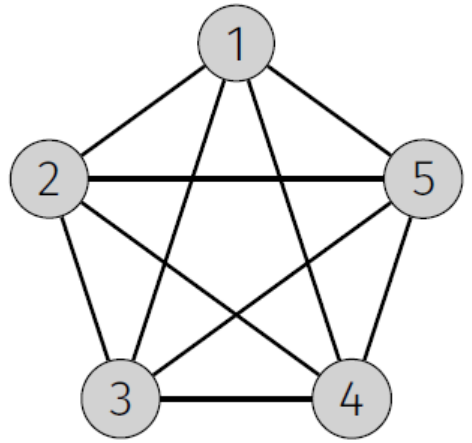
Eulerweg (Eulerian path):
Ein **Weg**, der jede Kante genau einmal benutzt (Endet **nicht** an gleichen Knoten)
→ *Nur wenn genau zwei Knoten mit ungeradem Grad*



Eulekreis (Eulerian cycle):
Ein **Kreis**, der jede Kante genau einmal benutzt (Endet an gleichen Knoten)
→ *Nur wenn alle Knoten geradem Grad*



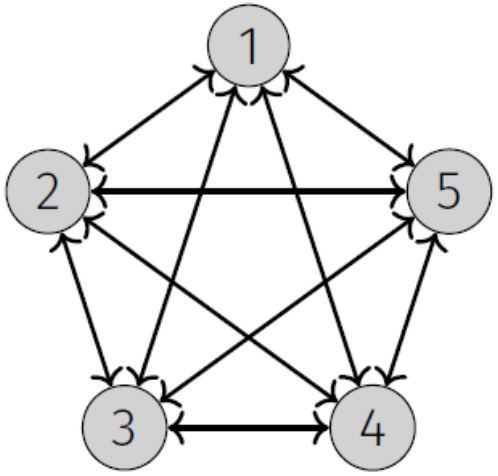
Handschlag-Lemma



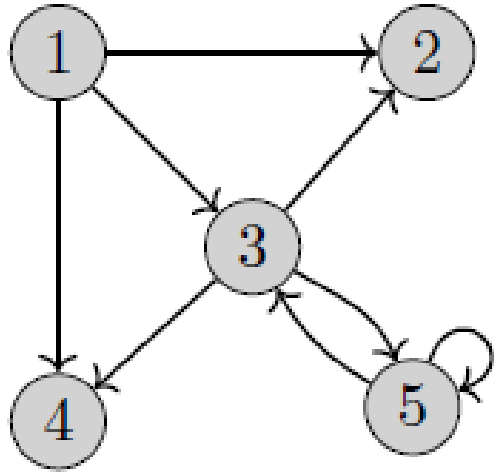
In jedem Graphen $G = (V, E)$ gilt

(a) $\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$, falls G gerichtet

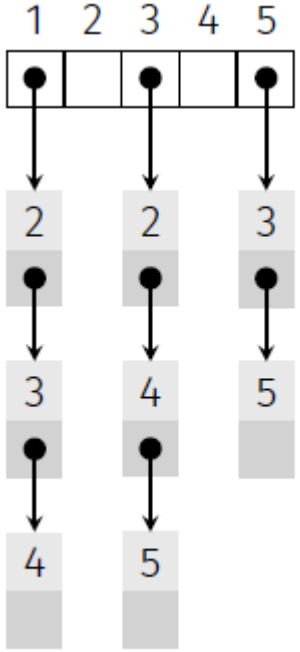
(b) $\sum_{v \in V} \deg(v) = 2|E|$, falls G ungerichtet.



Repräsentationsarten einer Matrix



$$A_G = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$



Adjazenzmatrix **Adjazenzliste**

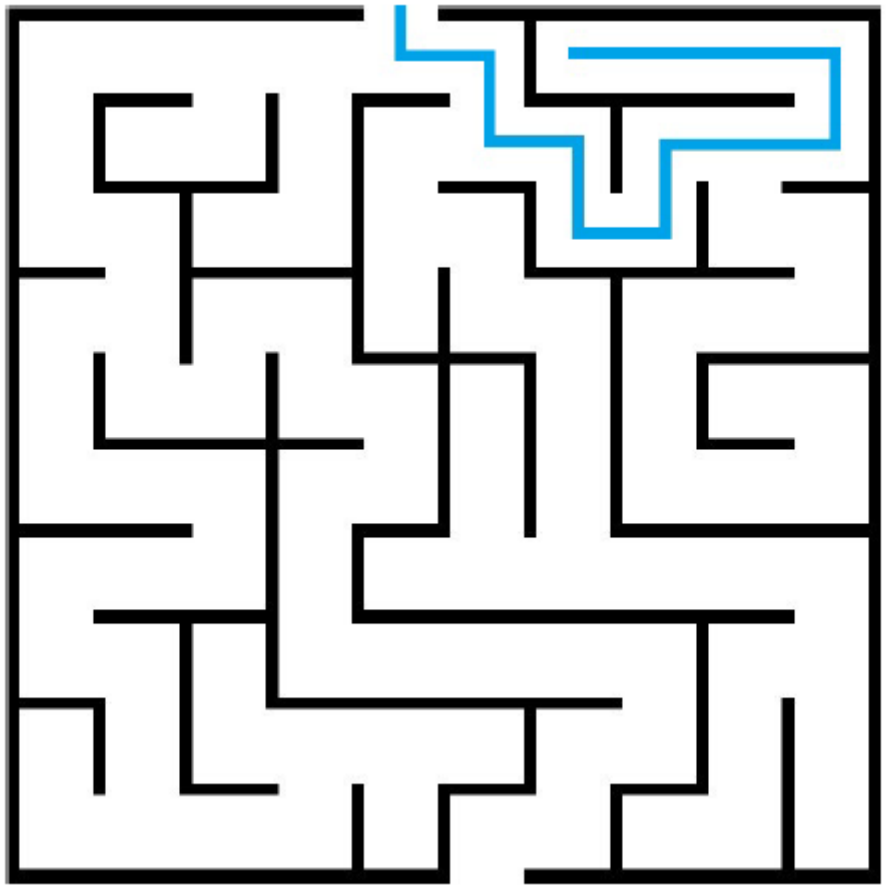
Übersicht der Darstellungsarten

Operation	Matrix	Liste
$(v,u) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	$\Theta(n)$
alle Kanten $e \in E$ finden	$\Theta(n^2)$	$\Theta(n+m)$
Kante einfügen	$\Theta(1)$	$\Theta(1)$
Kante löschen	$\Theta(1)$	$\Theta(\deg^+ v)$

n – Anzahl Knoten, m – Anzahl Kanten

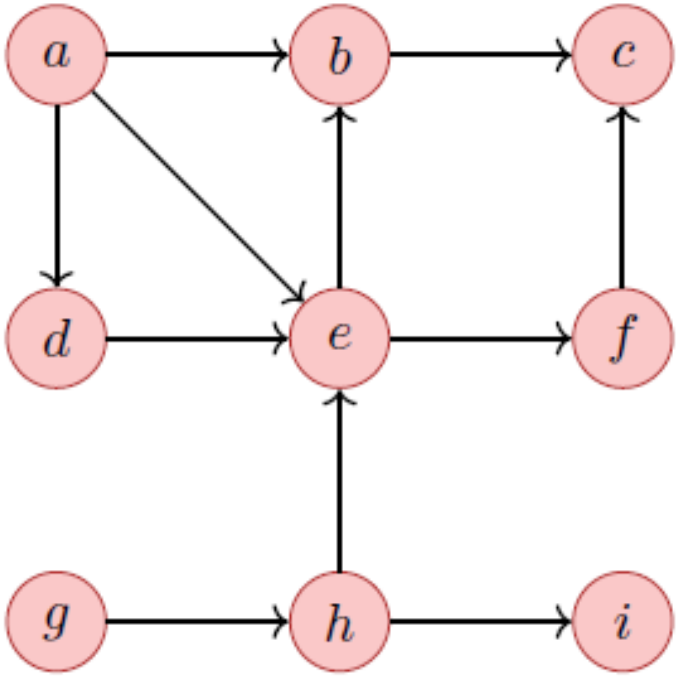
01 – Theorie
01.1 Graphen Definitionen
01.2 Tiefensuche
01.3 Breitensuche
01.4 Topologisches Sortieren

Tiefensuche (DFS)

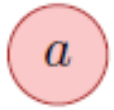


Tiefensuche (DFS)

DFS von a aus:



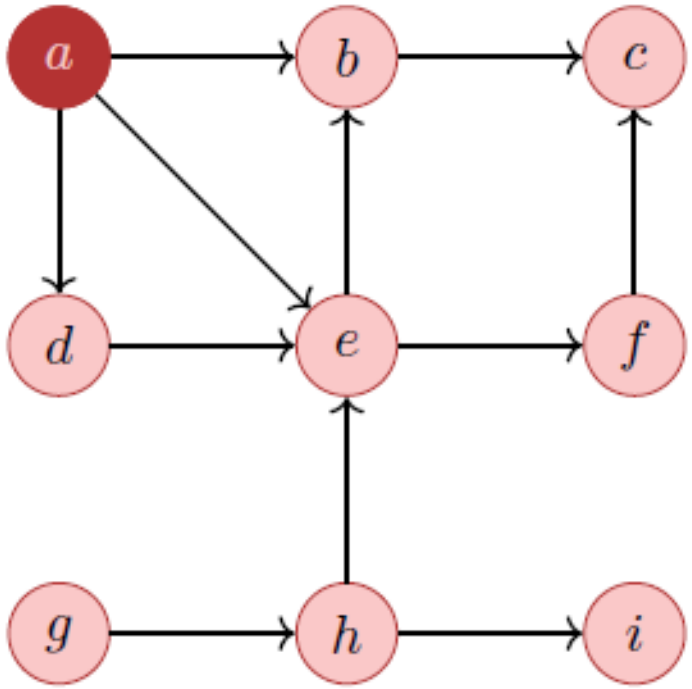
DFS-Baum: Distanzen und Vorgänger



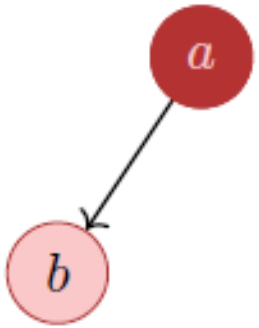
Distanz 0

Tiefensuche (DFS)

DFS von a aus:



DFS-Baum: Distanzen und Vorgänger

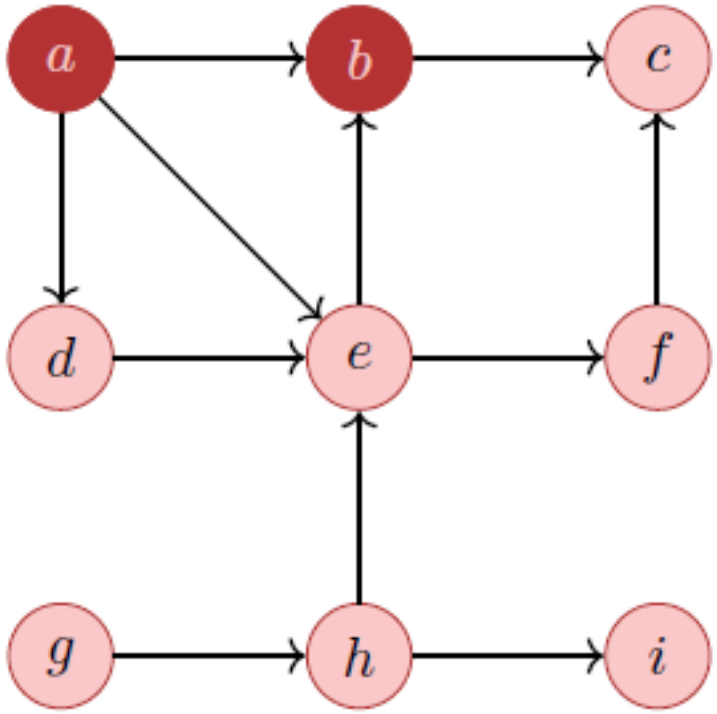


Distanz 0

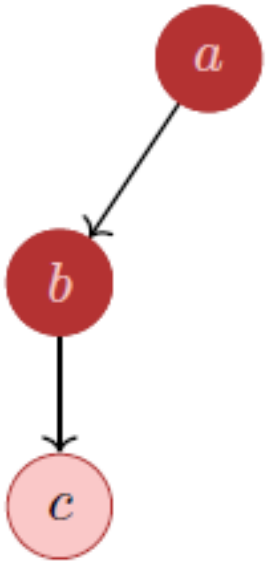
Distanz 1

Tiefensuche (DFS)

DFS von a aus:



DFS-Baum: Distanzen und Vorgänger



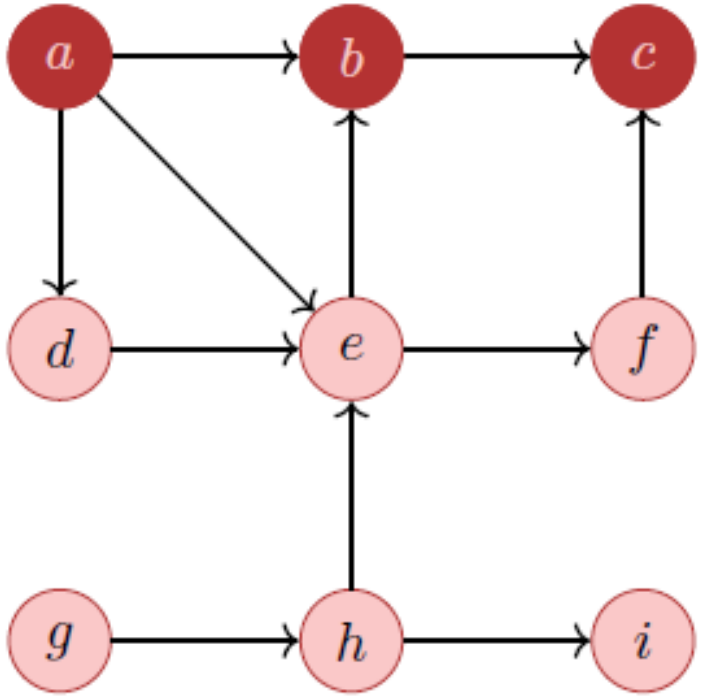
Distanz 0

Distanz 1

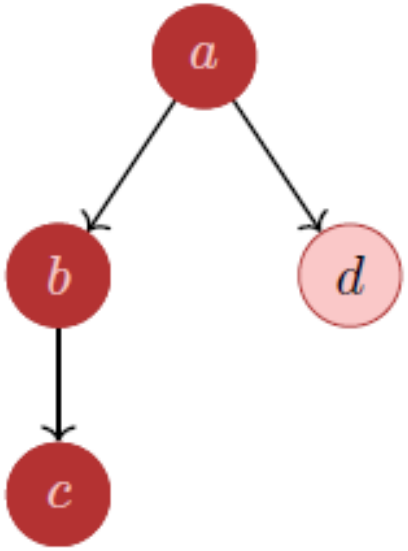
Distanz 2

Tiefensuche (DFS)

DFS von a aus:



DFS-Baum: Distanzen und Vorgänger



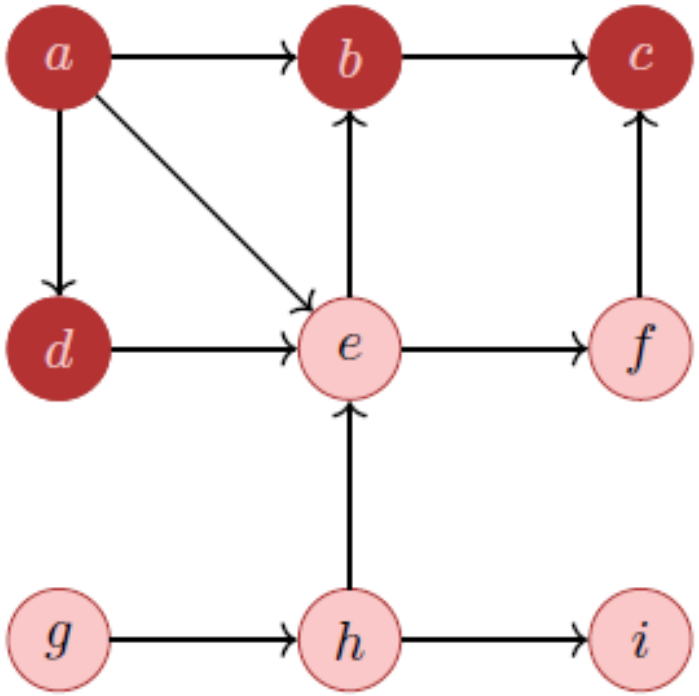
Distanz 0

Distanz 1

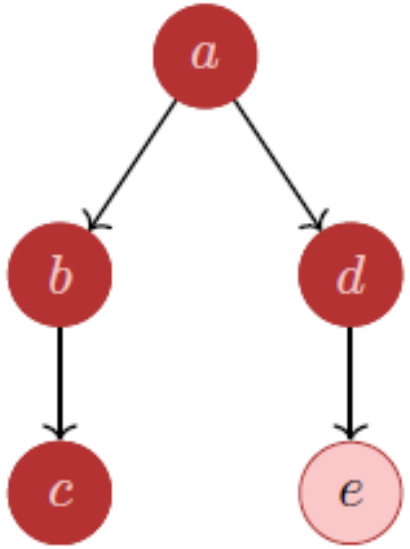
Distanz 2

Tiefensuche (DFS)

DFS von a aus:



DFS-Baum: Distanzen und Vorgänger



Distanz 0

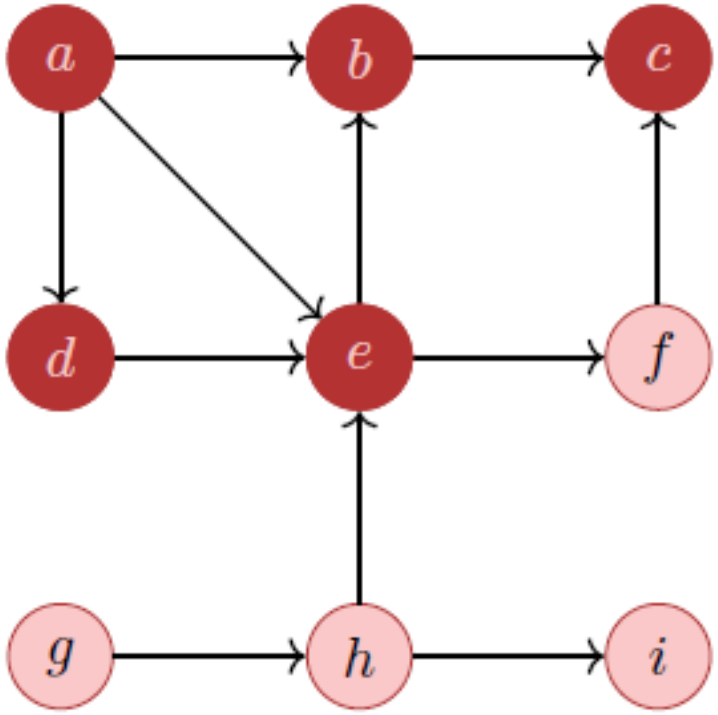
Distanz 1

Distanz 2

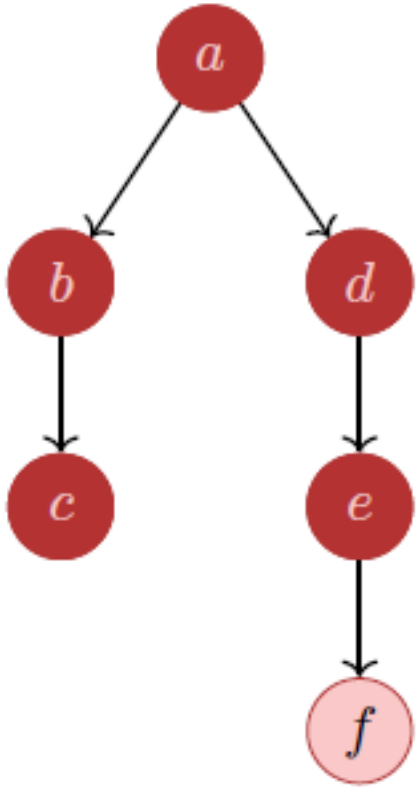
Distanz 3

Tiefensuche (DFS)

DFS von a aus:



DFS-Baum: Distanzen und Vorgänger



Distanz 0

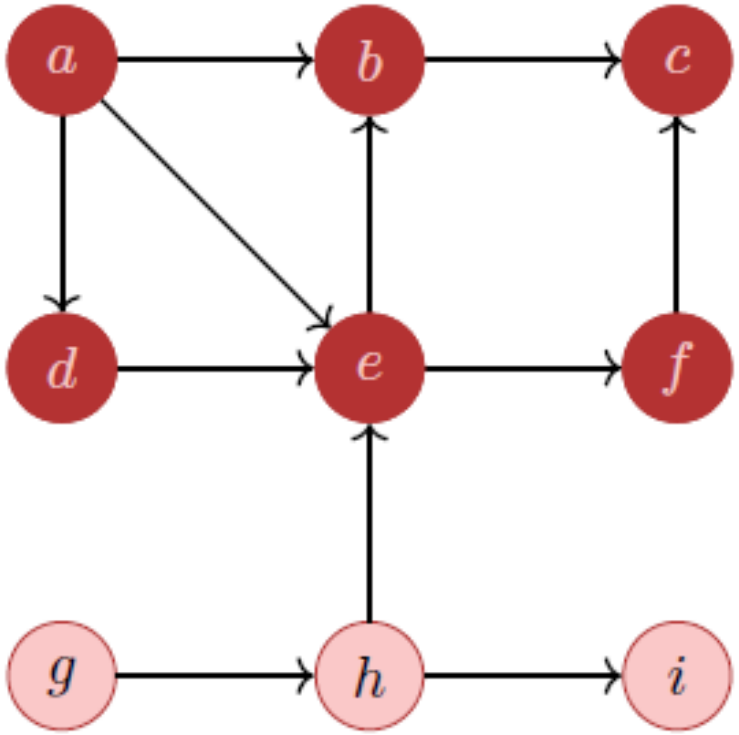
Distanz 1

Distanz 2

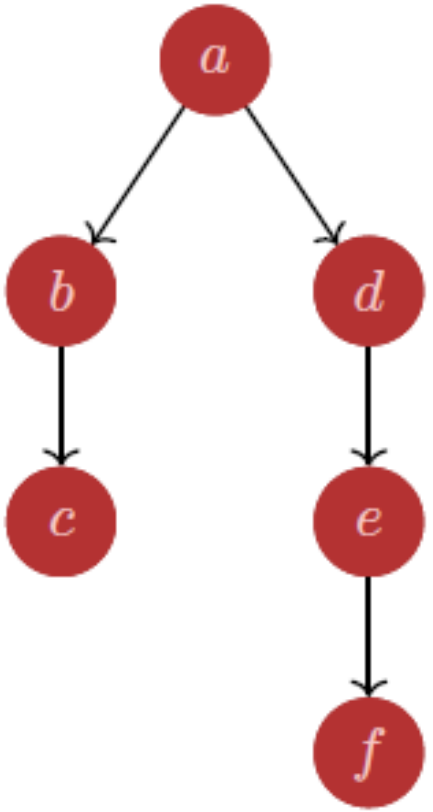
Distanz 3

Tiefensuche (DFS)

DFS von a aus:



DFS-Baum: Distanzen und Vorgänger



Distanz 0

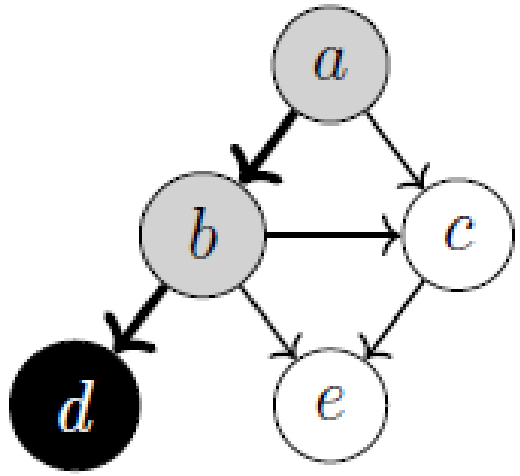
Distanz 1

Distanz 2

Distanz 3

Färben

Weiss: Knoten wurde noch nicht entdeckt.
Grau: Knoten wurde entdeckt ist in Bearbeitung.
Schwarz: Knoten wurde entdeckt und ist vollständig bearbeitet.



besucht: a, b, d
aktueller Knoten: b

DFS Algorithmus

Input: Graph $G = (V, E)$

foreach $v \in V$ **do**
    $v.color \leftarrow$ white

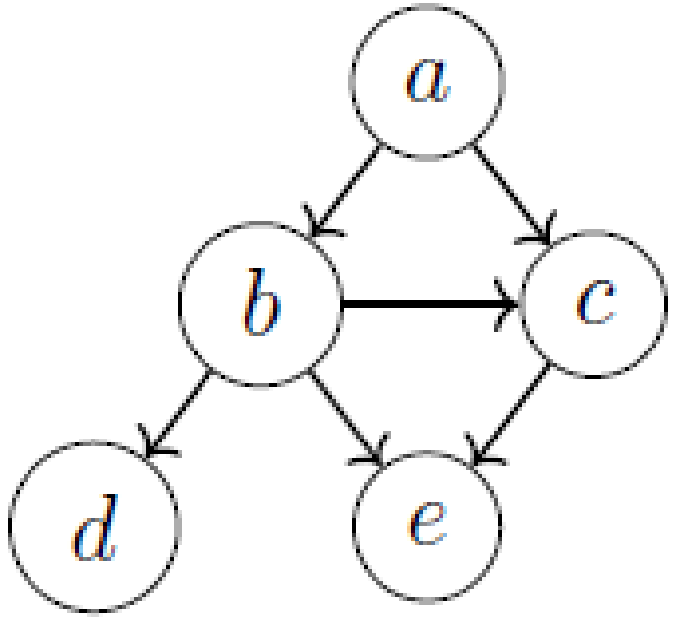
foreach $v \in V$ **do**
    if $v.color =$ white **then**
        DFS-Visit(G, v)

`DFS Visit(G, v):`

Input: Graph $G = (V, E)$, Knoten v.

$v.color \leftarrow$ grey
// besuche v
foreach $w \in N^+(v)$ **do**
    if $w.color =$ white **then**
        DFS-Visit(G, w)

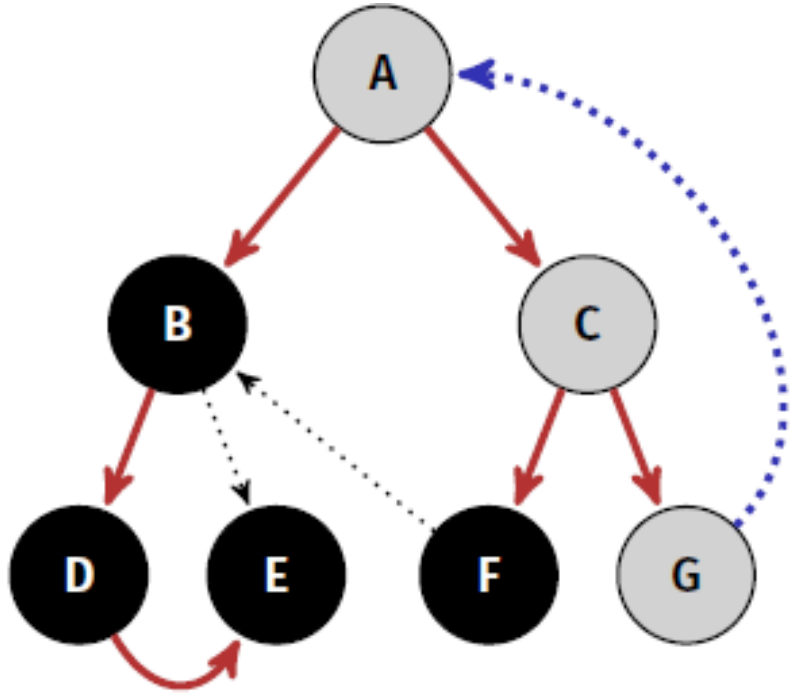
$v.color \leftarrow$ black



Laufzeit: $\Theta\left(\sum_{v \in V}(1 + \deg^+(v))\right) = \Theta(|V| + |E|)$

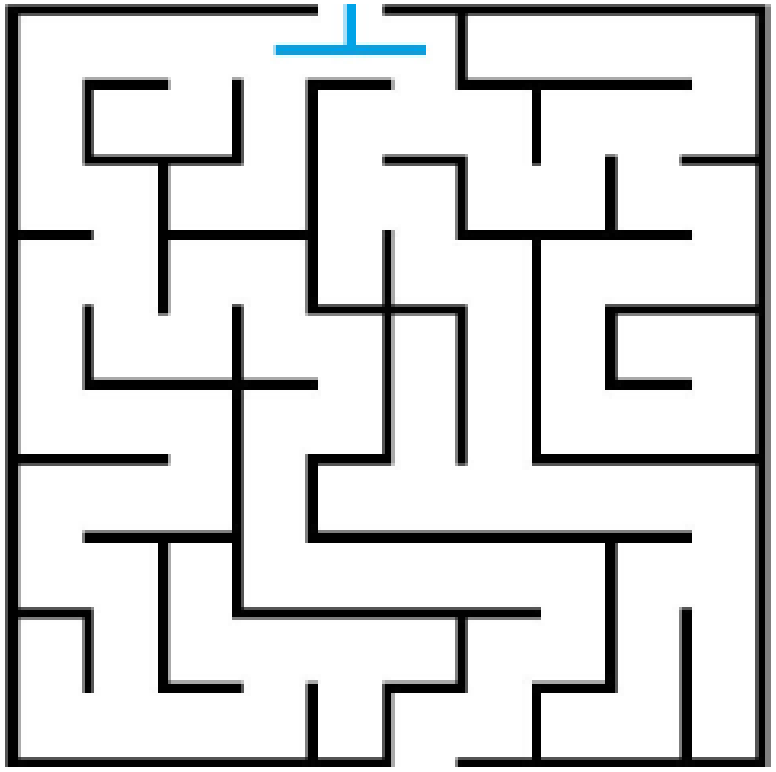
Kreis-Detektierung in DFS

Weiss/Graue Knoten dienen der Detektierung von Kreisen und Vermeidung dass der Algorithmus in Kreis lauft:



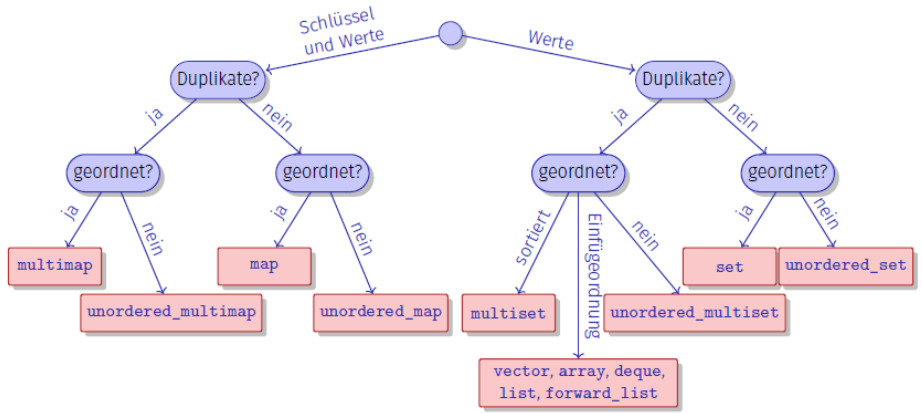
01 – Theorie
01.1 Graphen Definitionen
01.2 Tiefensuche
01.3 Breitensuche
01.4 Topologisches Sortieren

Breitensuche (BFS)



BFS

Datenstrukture Queue



std::queue

head → | 7 | 4 | 3 | 8 | 9 | 1 | 2 | ← tail

pop ()

output | 7 |

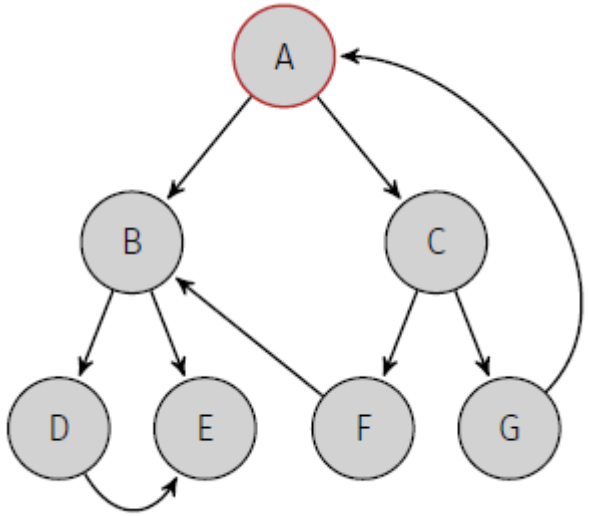
head → | 4 | 3 | 8 | 9 | 1 | 2 | ← tail

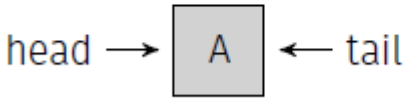
push(9)

head → | 9 | 4 | 3 | 8 | 9 | 1 | 2 | ← tail

https://en.cppreference.com/cpp/container/queue

Breitensuche (BFS)

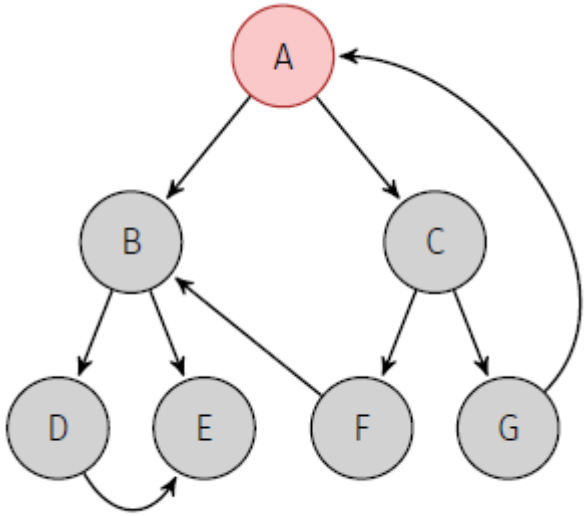




Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)



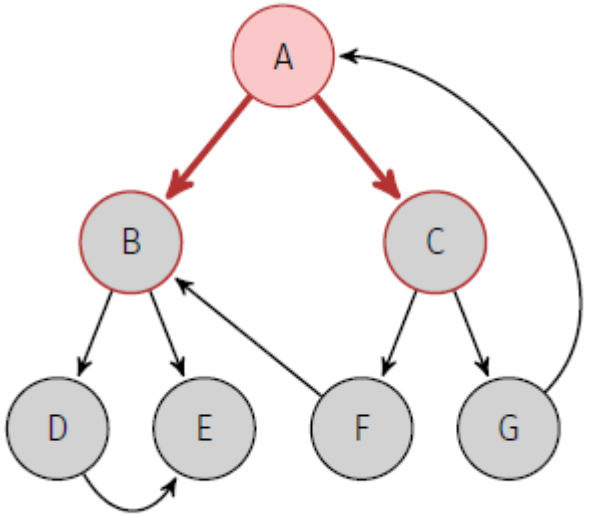
$v = A$

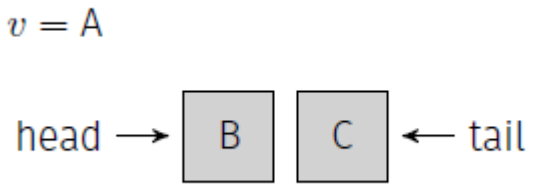
head ⟶ ⟵ tail

Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)

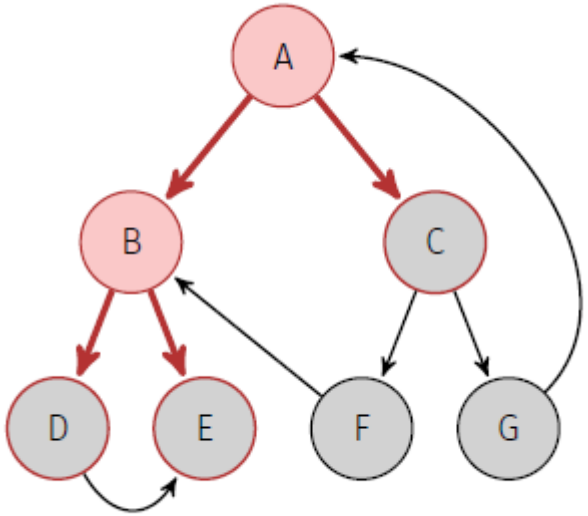




Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)



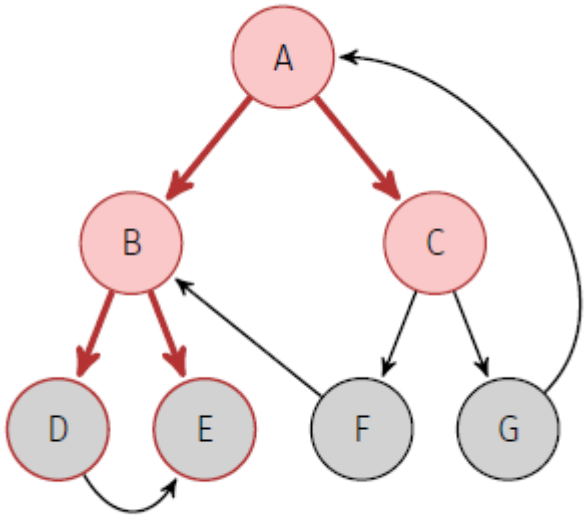
$v = B$



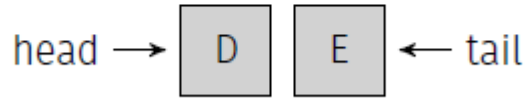
Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)



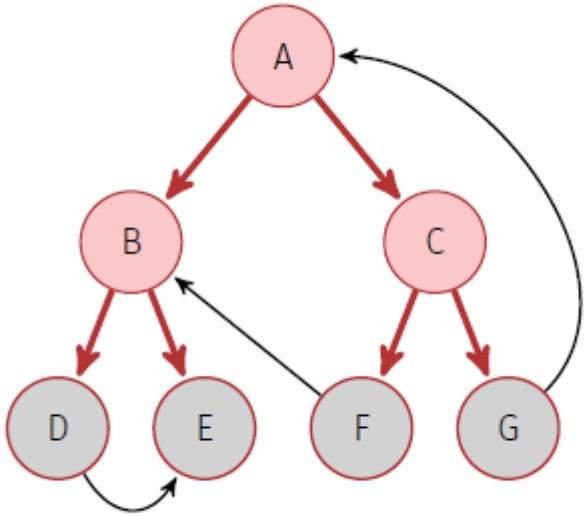
$v = C$



Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)



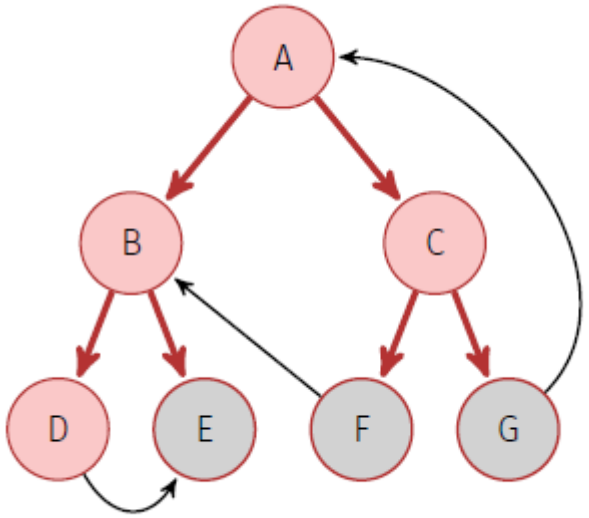
$v = C$

Besuchter Knoten



Warteschlange (queue)

Breitensuche (BFS)

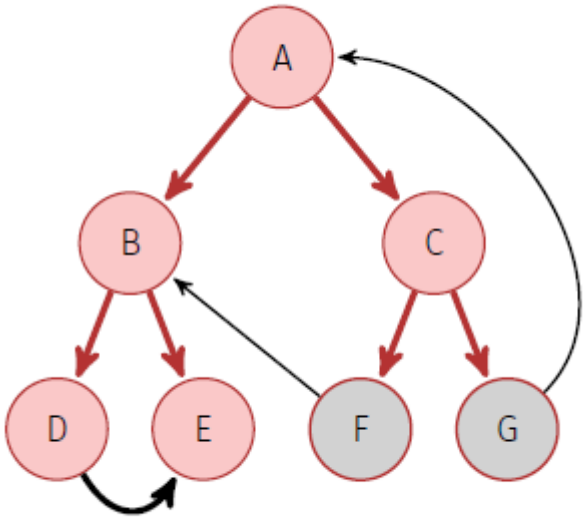


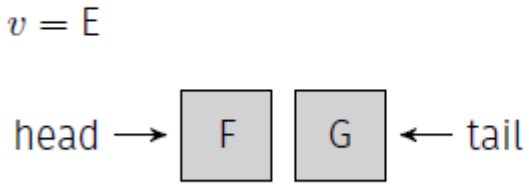


Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)

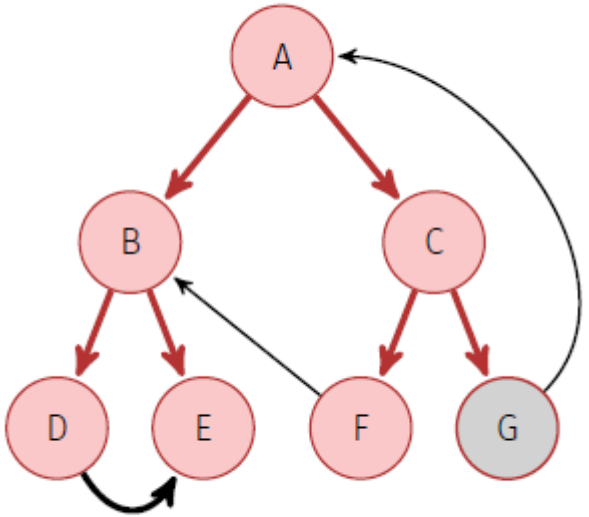




Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)



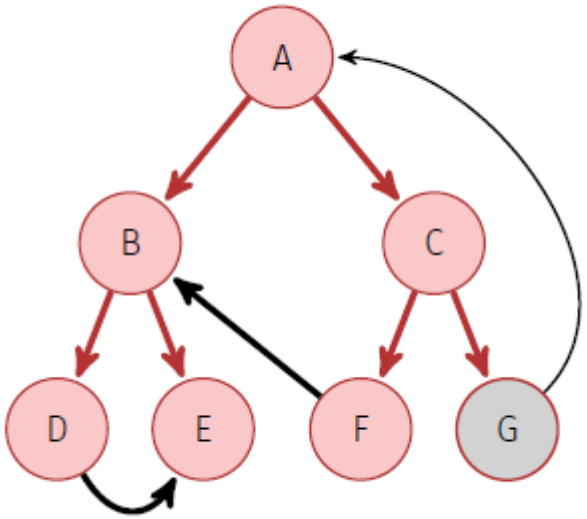
$v = \mathrm{F}$

head → G ← tail

Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)



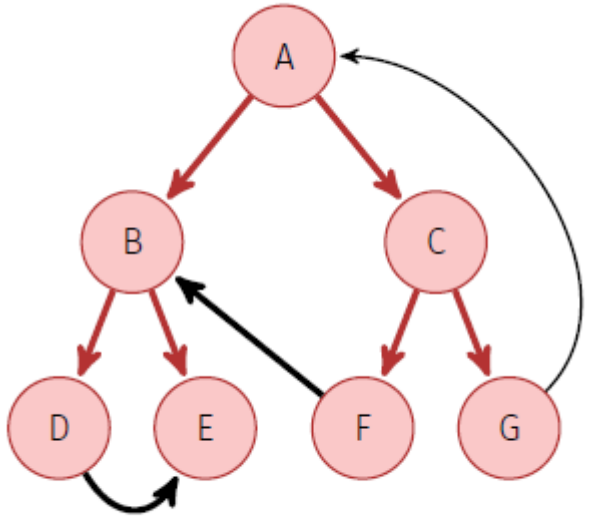
$v = F$

head → G ← tail

Besuchter Knoten

Warteschlange (queue)

Breitensuche (BFS)



$v = \mathrm{G}$

head → ← tail

Besuchter Knoten

Warteschlange (queue)

Breitensuche Algoritmus

Input: Graph $G = (V, E)$

foreach $v \in V$ **do**
    v.seen $\leftarrow$ false

foreach $v \in V$ **do**
    if v.seen $=$ false **then**
        BFS-Visit(G,v)

Laufzeit: $\Theta(|V| + |E|)$

`BFS Visit(G, v)`

Input: Graph $G = (V, E)$, Knoten $v \in V$

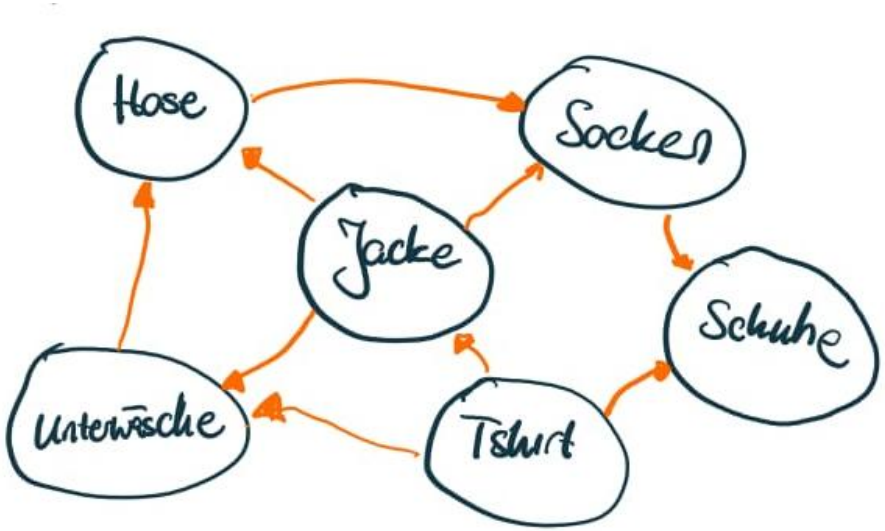
Queue $Q \leftarrow \emptyset$
enqueue(Q, v)
v.seen $\leftarrow$ true
while $Q \neq \emptyset$ **do**
    $w \leftarrow$ dequeue(Q)
    // besuche w
    foreach $c \in N^+(w)$ **do**
        if c.seen $=$ false **then**
            c.seen $\leftarrow$ true
            enqueue(Q, c)

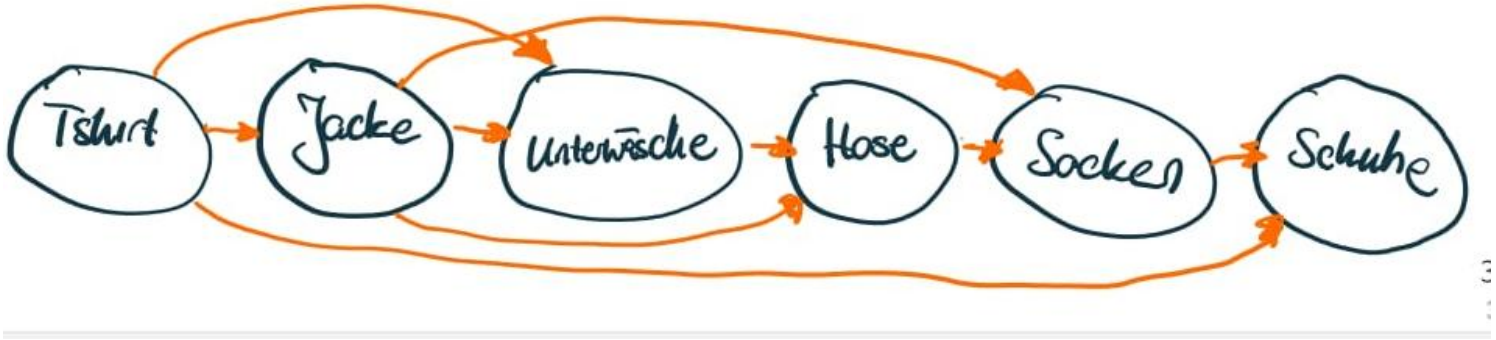
01 – Theorie
01.1 Graphen Definitionen
01.2 Tiefensuche
01.3 Breitensuche
01.4 Topologisches Sortieren

Topologisches Sortieren

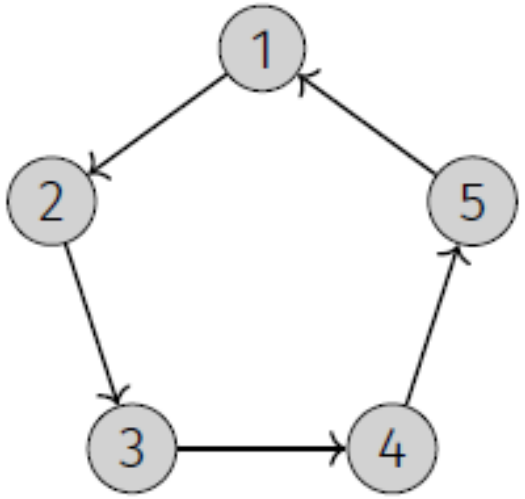
Stell dir vor du willst entscheiden in Welcher Reihenfolge du dich anziehen willst.

Gewisse Kleidungsstücke müssen vorher oder nach einem anderen Kleidungsstück angezogen werden....



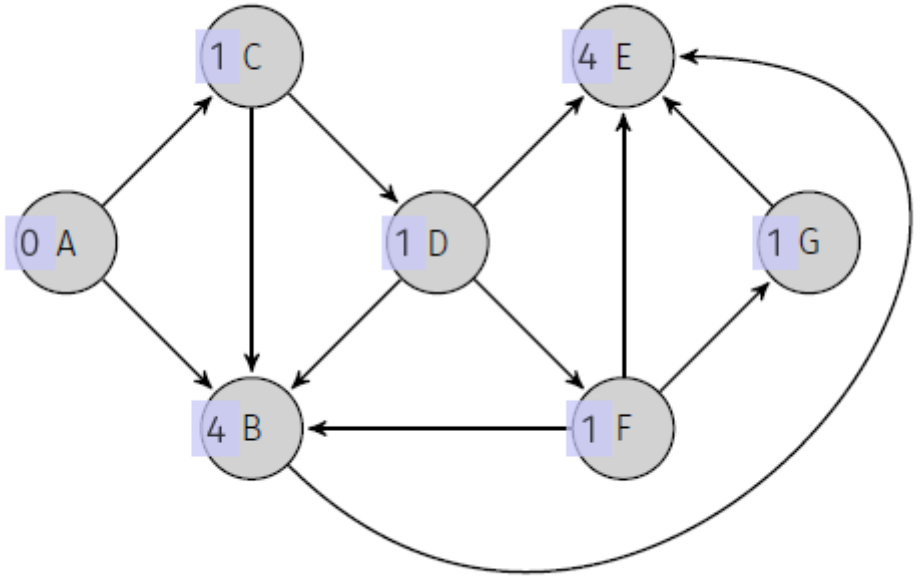


Topologisches Sortieren



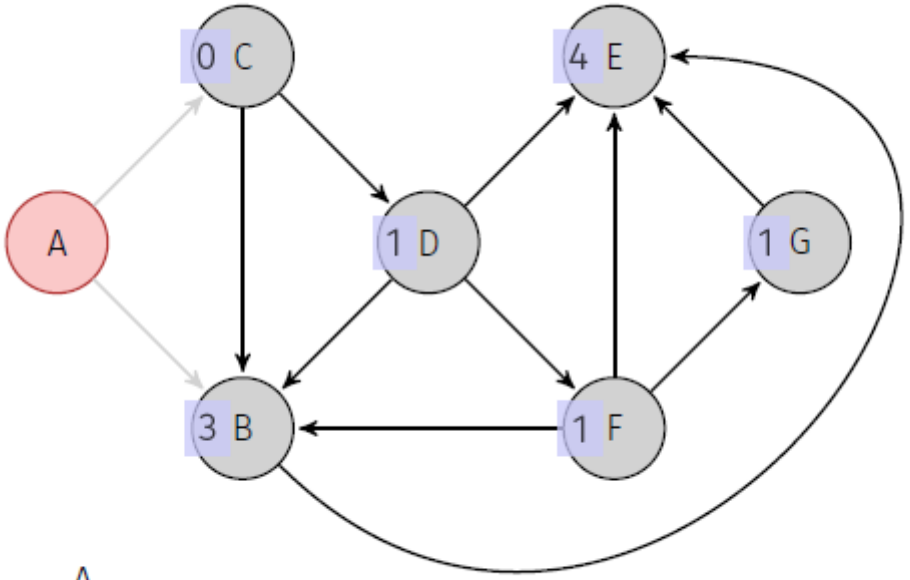
Zyklischer Graph (Graph mit Kreis): kann nicht topologisch sortiert werden.

Topologisches Sortieren



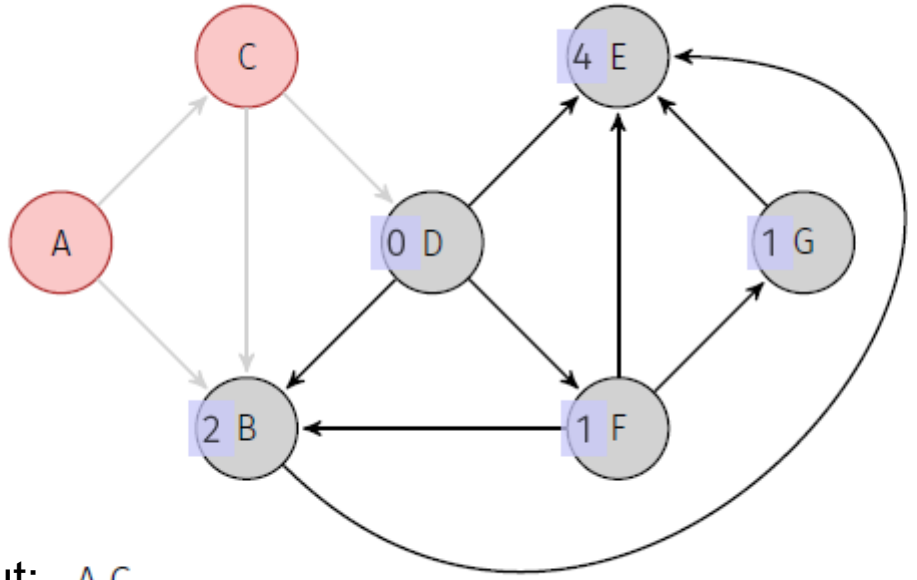
Output:

Topologisches Sortieren



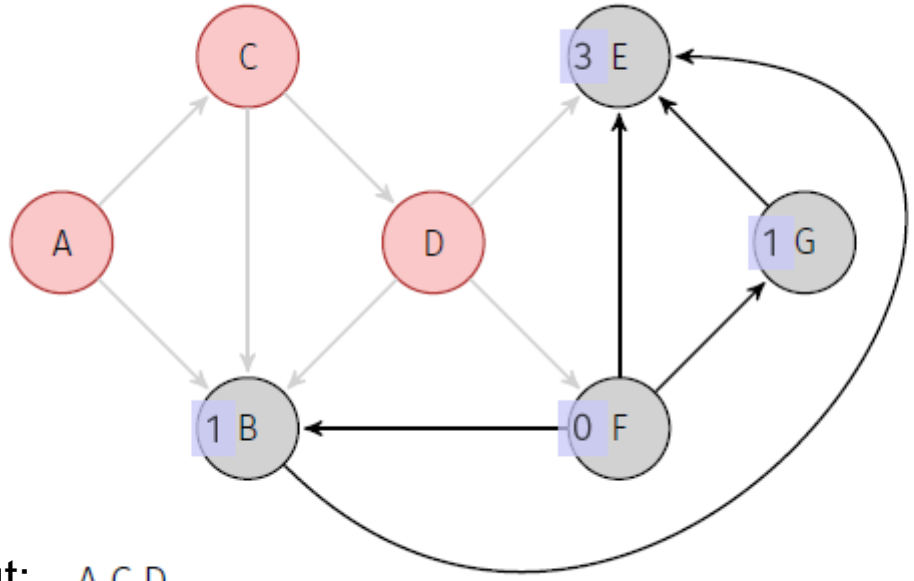
Output: A

Topologisches Sortieren



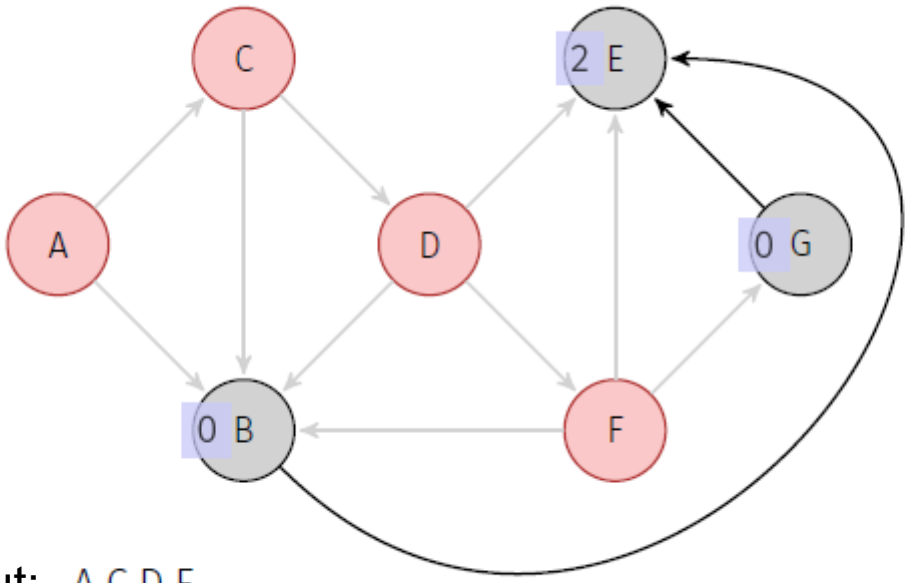
Output: A C

Topologisches Sortieren



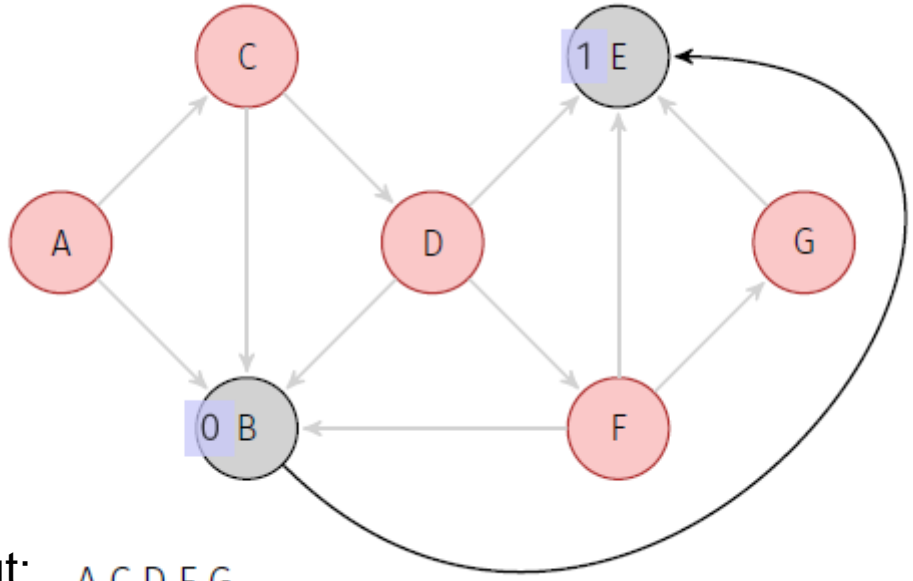
Output: A C D

Topologisches Sortieren



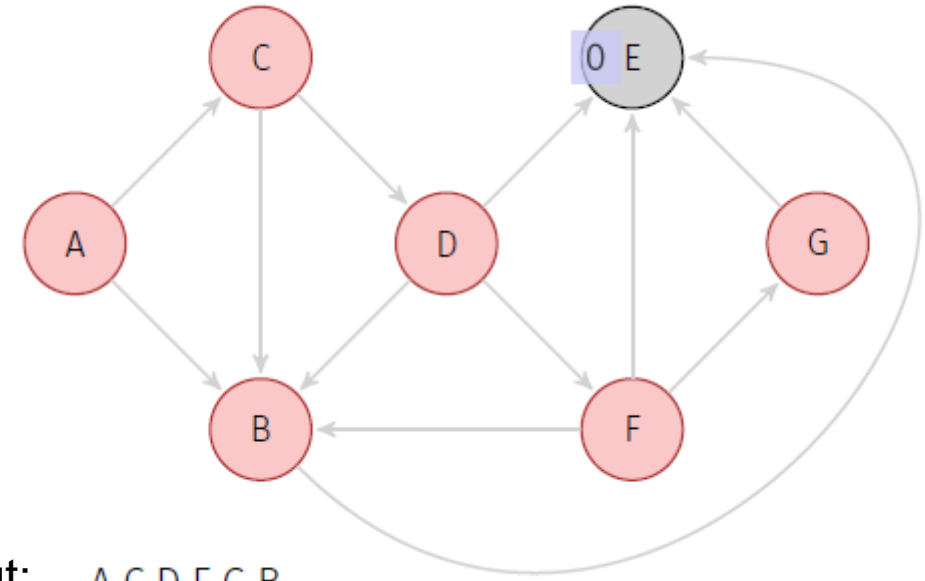
Output: A C D F

Topologisches Sortieren



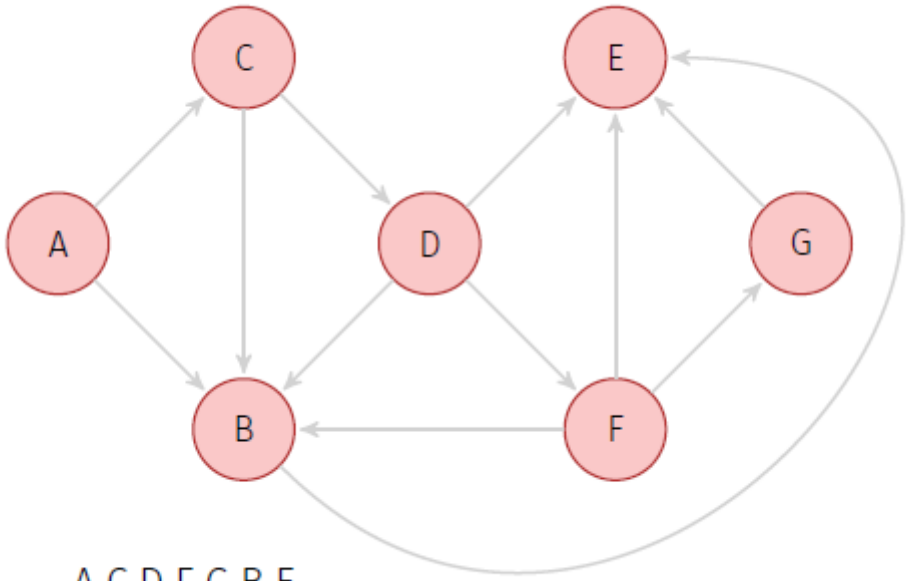
Output: A C D F G

Topologisches Sortieren



Output: A C D F G B

Topologisches Sortieren



Output: A C D F G B E

Topologisches Sortieren Algorithmus

Input: Graph $G = (V, E)$.
Output: Topologische Sortierung ord

Stack $S \leftarrow \emptyset$
foreach $v \in V$ **do** $A[v] \leftarrow 0$
foreach $(v, w) \in E$ **do** $A[w] \leftarrow A[w] + 1$ // Eingangsgrade berechnen
foreach $v \in V$ with $A[v] = 0$ **do** push(S, v) // Merke Nodes mit Eingangsgrad 0
$i \leftarrow 1$
while $S \neq \emptyset$ **do**
    $v \leftarrow$ pop(S); ord$[v] \leftarrow i$; $i \leftarrow i + 1$ // Wähle Knoten mit Eingangsgrad 0
    foreach $w \in N^+(v)$ **do** // Verringere Eingangsgrad der Nachfolger
        $A[w] \leftarrow A[w] - 1$
        if $A[w] = 0$ **then** push(S, w)

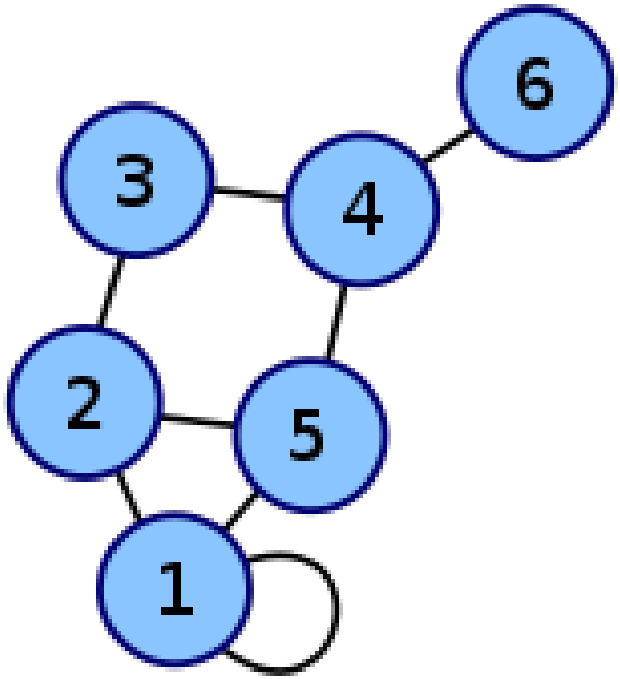
if $i = |V| + 1$ **then return** ord **else return** "Cycle Detected"

02 – Quiz

Quiz Frage 1

Was ist die Summe von Kantengrade in diesem Graph?

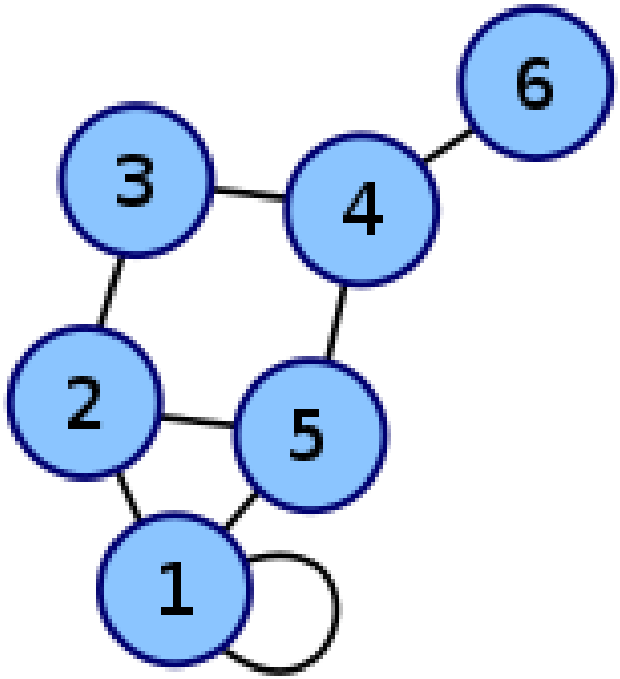
a) 9
b) 15
c) 16
d) 32



Quiz Frage 1

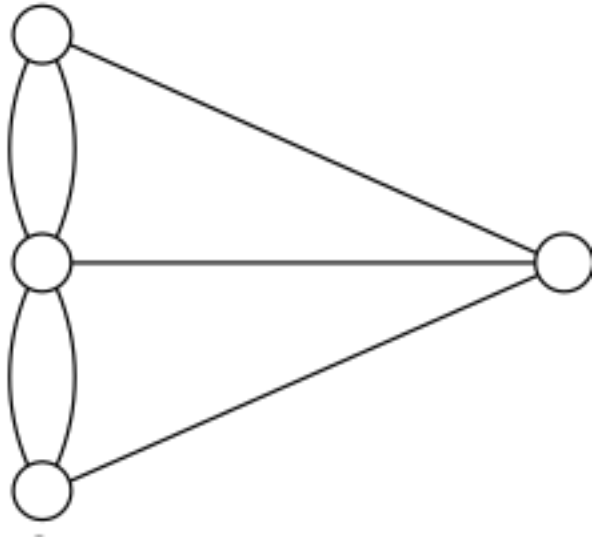
Was ist die Summe von Kantengrade in diesem Graph?

a) 9
b) 15
c) 16
d) 32



Quiz Frage 2

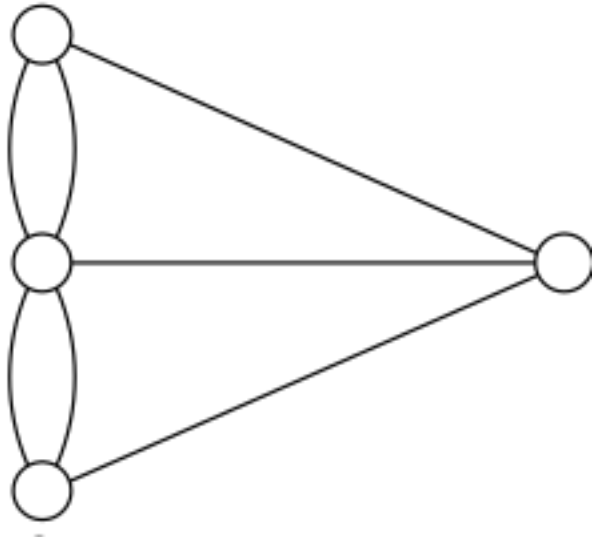
Welche Aussage über Besuche von Kanten trifft zu:



a) Der Graph hat einen Eulerkreis
b) Der Graph hat einen Eulerweg
c) Der Graph hat sowohl einen Eulerweg wie einen Eulerkreis
d) Keine davon

Quiz Frage 2

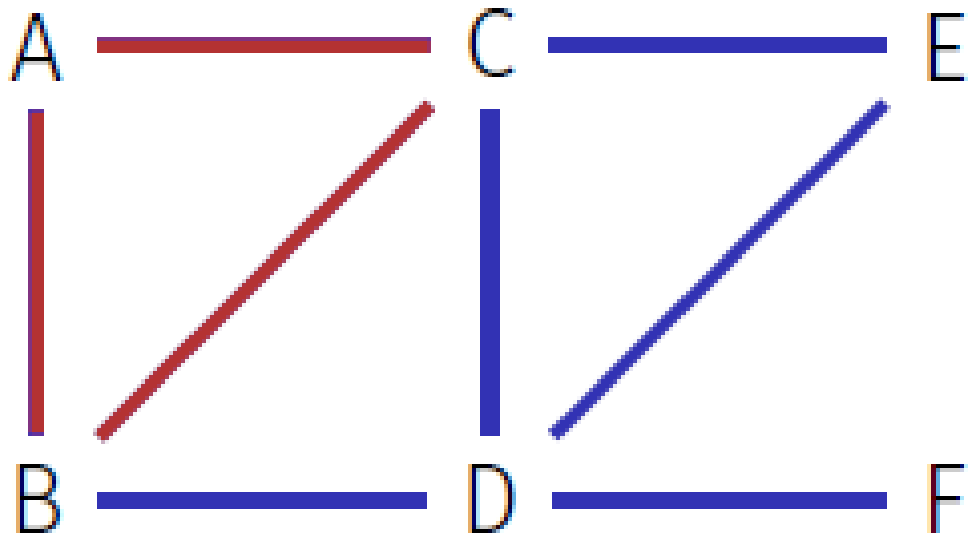
Welche Aussage über Besuche von Kanten trifft zu:



a) Der Graph hat einen Eulerkreis
b) Der Graph hat einen Eulerweg
c) Der Graph hat sowohl einen Eulerweg wie einen Eulerkreis
d) Keine davon

Quiz Frage 3

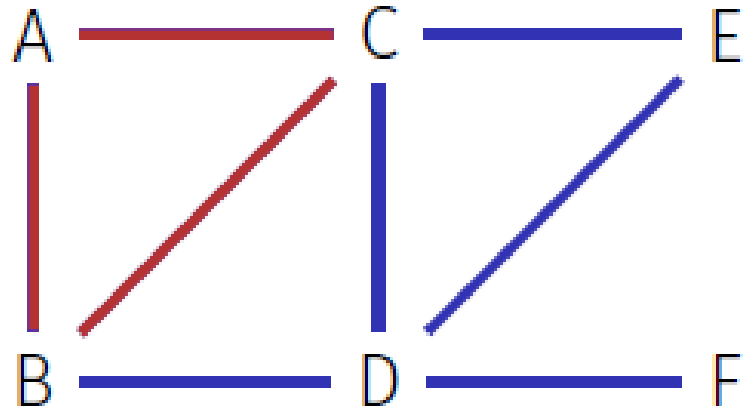
Der Graph wird mit einer Adjazenzmatrix dargestellt. Wie viele Operationen brauchen wir zum sicherzustellen, dass das Graph zusammenhängend ist?



a) 4
b) 8
c) 16
d) 36

Quiz Frage 3

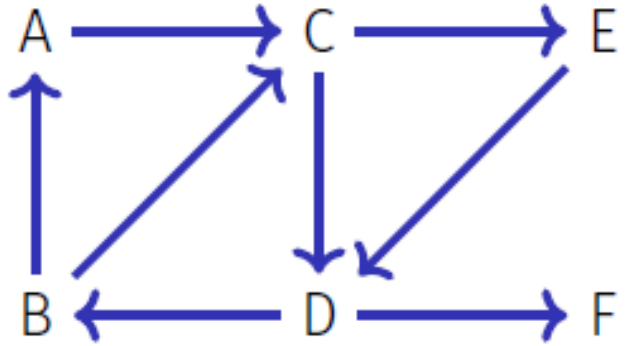
Der Graph wird mit einer Adjazenzmatrix dargestellt. Wie viele Operationen brauchen wir zum sicherzustellen, dass das Graph zusammenhängend ist?



a) 4
b) 8
c) 16
d) 36

Quiz Frage 4

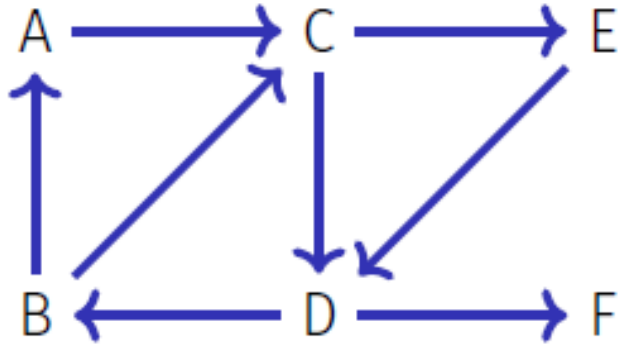
Der folgende Graph wird, ausgehend vom Knoten *A* aus mit **Breitensuche** besucht. Wenn es mehrere Möglichkeiten für eine Besuchsreihenfolge der Nachbarn gibt, wird die alphabetische Ordnung genommen.



a) A C D E B F
b) A C E D B F
c) A C D B F E
d) A C D F B E

Quiz Frage 4

Der folgende Graph wird, ausgehend vom Knoten *A* aus mit **Breitensuche** besucht. Wenn es mehrere Möglichkeiten für eine Besuchsreihenfolge der Nachbarn gibt, wird die alphabetische Ordnung genommen.

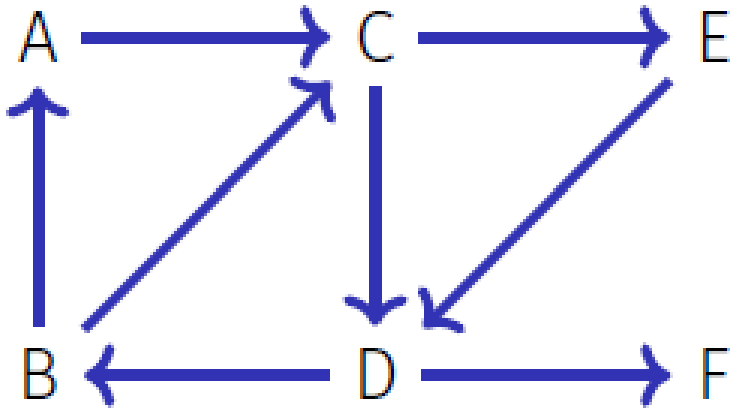


a) A C D E B F
b) A C E D B F
c) A C D B F E
d) A C D F B E

Quiz Frage 5

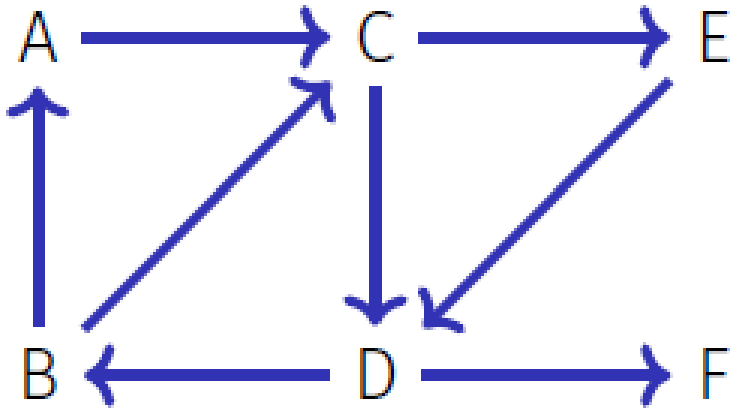
Der folgende Graph wird, ausgehend vom Knoten *A* aus mit **Tiefensuche** besucht. Wenn es mehrere Möglichkeiten für eine Besuchsreihenfolge der Nachbarn gibt, wird die alphabetische Ordnung genommen.

a) A C D E B F
b) A C E D B F
c) A C D B F E
d) A C D F B E



Quiz Frage 5

Der folgende Graph wird, ausgehend vom Knoten *A* aus mit **Tiefensuche** besucht. Wenn es mehrere Möglichkeiten für eine Besuchsreihenfolge der Nachbarn gibt, wird die alphabetische Ordnung genommen.

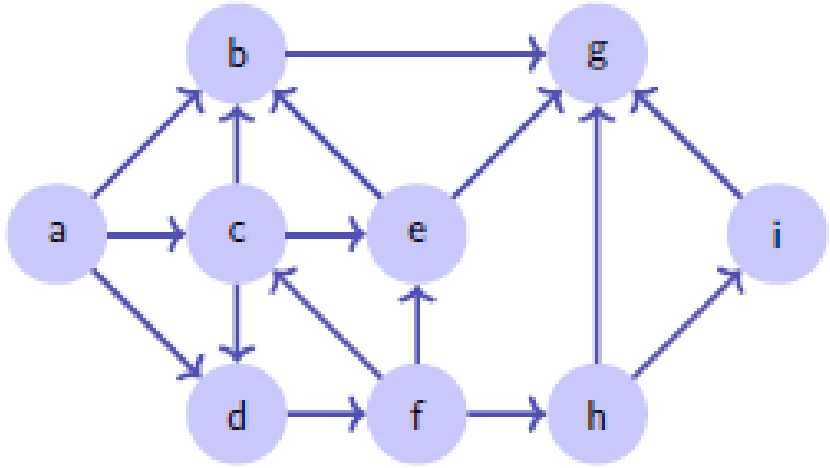


a) A C D E B F
b) A C E D B F
c) A C D B F E
d) A C D F B E

Quiz Frage 6 (HS 24)

Der Graph wird mit der **Breitensuche** besucht mit Startknoten a. Welche Traversierung ist möglich?

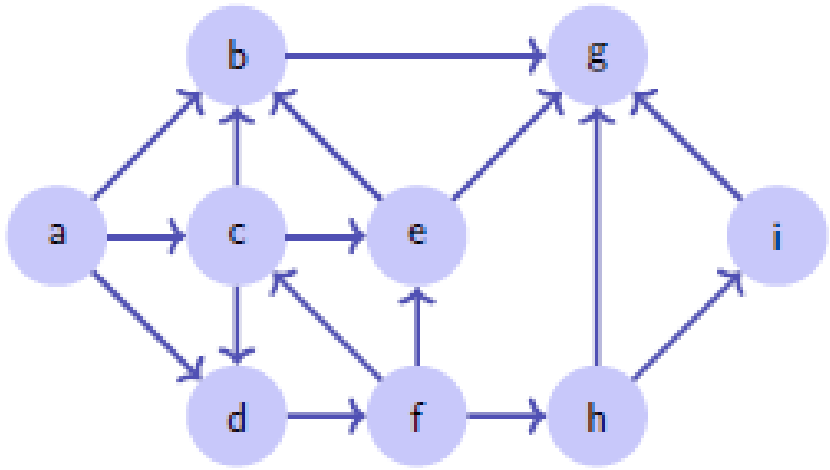
a) a d f c e b g h i
b) a d c e b f g i h
c) a d c b f e g h i
d) a b g c e d f h i



Quiz Frage 6 (HS 24)

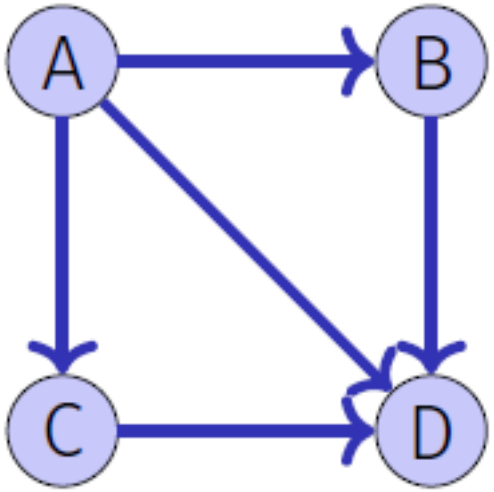
Der Graph wird mit der **Breitensuche** besucht mit Startknoten a. Welche Traversierung ist möglich?

a) a d f c e b g h i
b) a d c e b f g i h
c) a d c b f e g h i
d) a b g c e d f h i



Quiz Frage 7

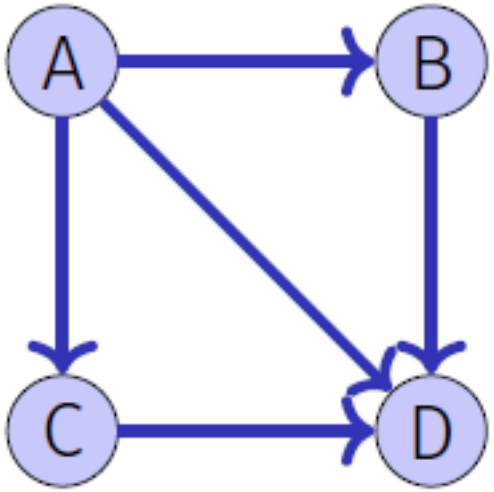
Auf wie viele Arten kann man den folgenden Graphen topologisch sortieren?



a) 0
b) 1
c) 2
d) 3

Quiz Frage 7

Auf wie viele Arten kann man den folgenden Graphen topologisch sortieren?

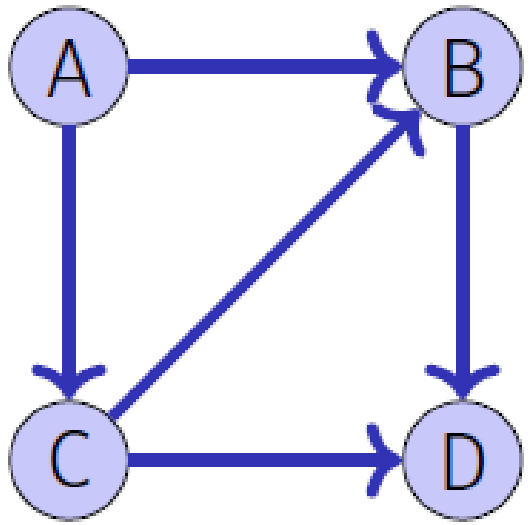


a) 0
b) 1
c) 2
d) 3

Quiz Frage 8

Auf wie viele Arten kann man den folgenden Graphen topologisch sortieren?

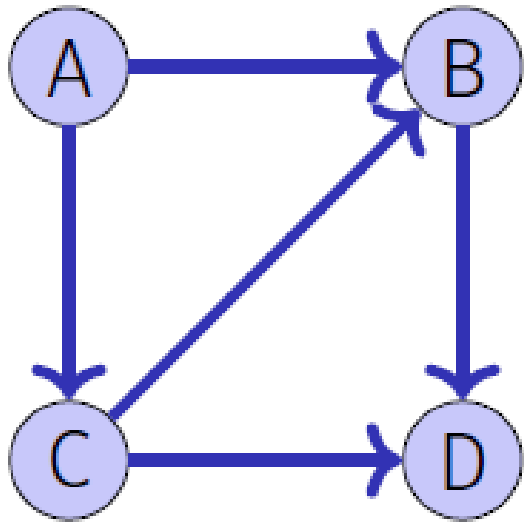
a) 0
b) 1
c) 2
d) 3



Quiz Frage 8

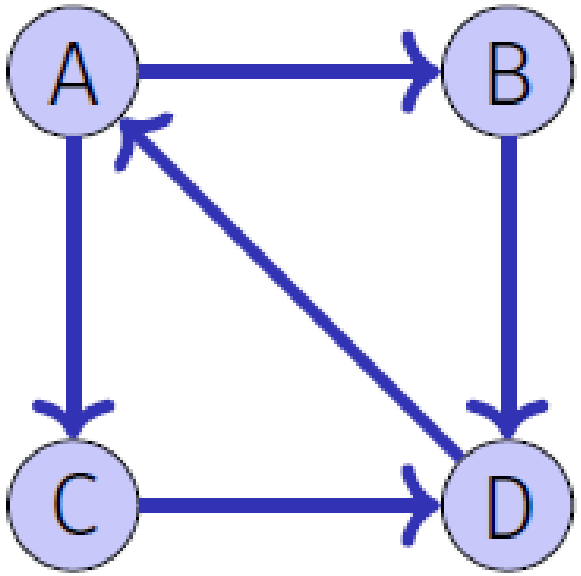
Auf wie viele Arten kann man den folgenden Graphen topologisch sortieren?

a) 0
b) 1
c) 2
d) 3



Quiz Frage 9

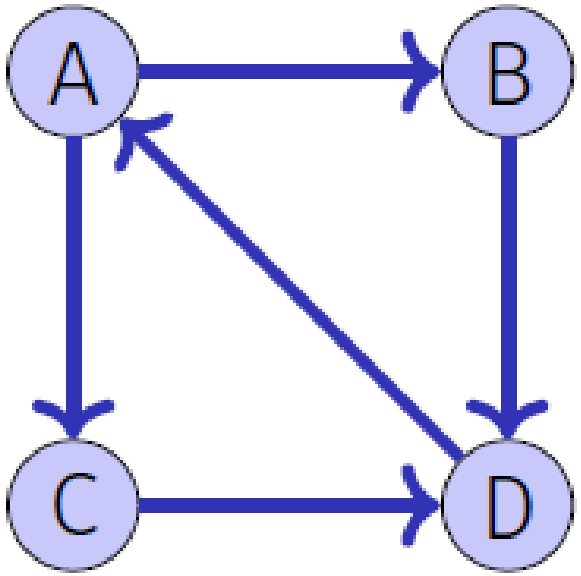
Auf wie viele Arten kann man den folgenden Graphen topologisch sortieren?



a) 0
b) 1
c) 2
d) 3

Quiz Frage 9

Auf wie viele Arten kann man den folgenden Graphen topologisch sortieren?

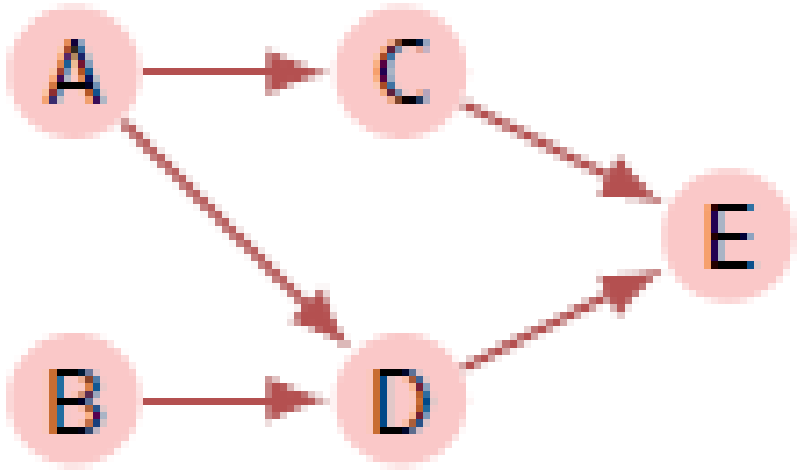


a) 0
b) 1
c) 2
d) 3

Quiz Frage 10 (FS24)

Welche der folgenden Sequenzen ist eine topologiesche Sortierung?

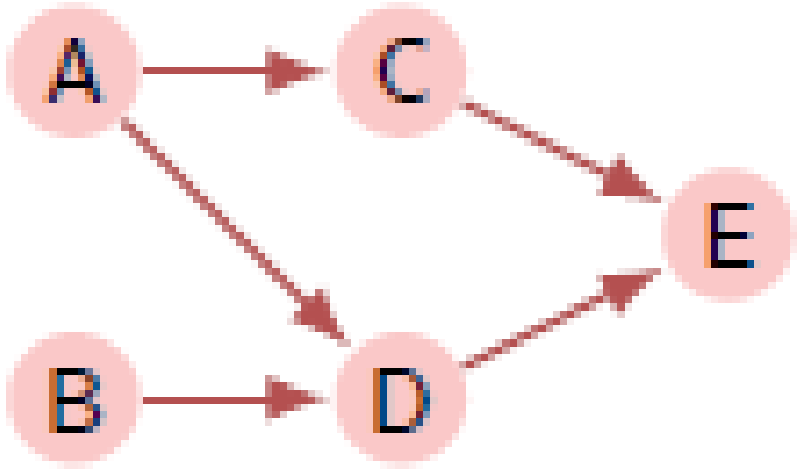
a) ABCED
b) ACBED
c) ABDCE
d) BDACE



Quiz Frage 10 (FS24)

Welche der folgenden Sequenzen ist eine topologiesche Sortierung?

a) ABCED
b) ACBED
c) ABDCE
d) BDACE



03 – Programmierübung

Programmierübung: BFS Tree

```cpp
#include <iostream>
#include <deque>
#include "node.h"

// Function declarations
void DFS(Node* node);
void BFS(Node* node);

// Function implementations

void DFS(Node* node){
  if (node != nullptr){
    std::cout << node->key << " ";
    DFS(node -> left);
    DFS(node -> right);
  }
}

// implement this: breadth first search: output the key values of all nodes
// level by level (root first, then its children, then children of children, etc.)
void BFS(Node* node){

  // implement me

}
```

1. Geh auf CodeExpert → Code Examples → BFS Tree
2. Implementiere `BFS(Node* node)`, das ein Baum mit Breitensuche traversiert

```
Input: 1 2 3 4 5 6
Output:
        ┌─ 4
    ┌─ 2
    │   └─ 5
── 1
    │   ┌─ 6
    └─ 3

DFS: 1 2 4 5 3 6 BFS: 1 2 3 4 5 6
```

ETH zürich

Michal Oskedra
Teaching Assistent D-ITET
moskedra@ethz.ch