7 Graphen

Formale Grundlagen der Informatik I

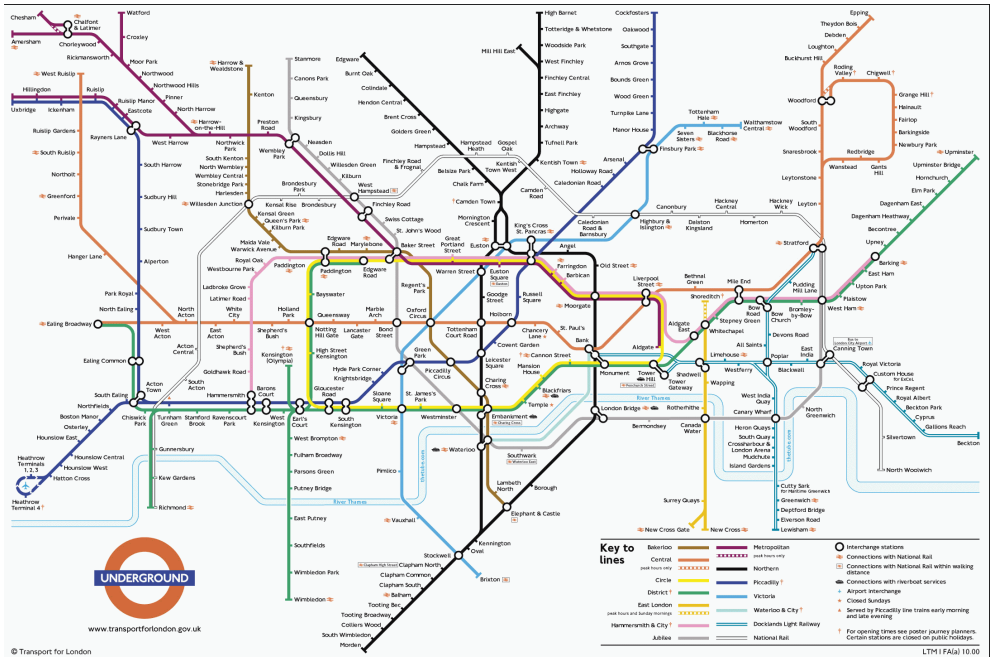
Herbstsemester 2012

Robert Marti

Vorlesung teilweise basierend auf Unterlagen
von Prof. emer. Helmut Schauer

Graphen als Abstraktion von Netzwerken

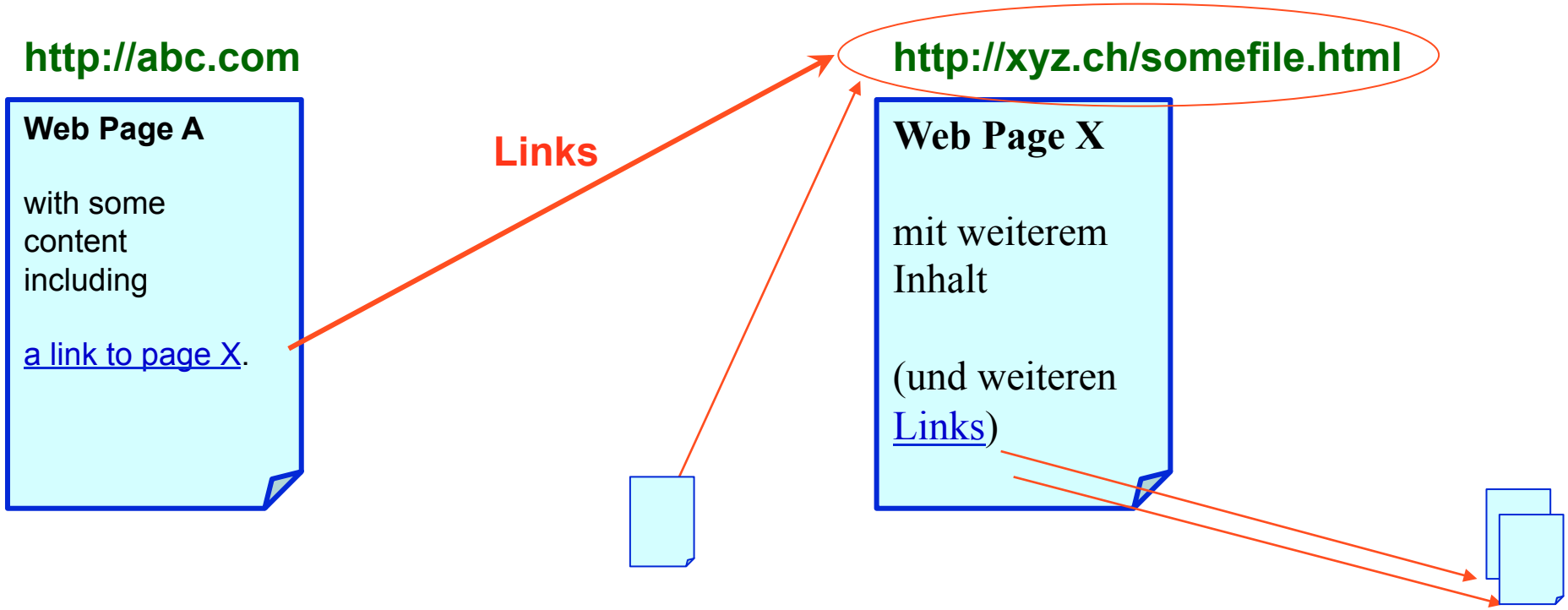
- Interesse an Fragen wie
  - Existenz von Verbindungen
  - Existenz von Zyklen
  - Distanzen, Zeiten, Durchsatz, …



- Anwendungen
  - Verkehrsnetze / Fahrpläne
  - Abhängigkeitsgraphen, z.B. "cause-effect", Arbeitspläne, Lehrpläne
  - Physische Computernetzwerke (z.B. Maschinen als Knoten)
  - Logische Computernetzwerke (z.B. Web Seiten als Knoten)
  - soziale Netzwerke (bis hin zu facebook)

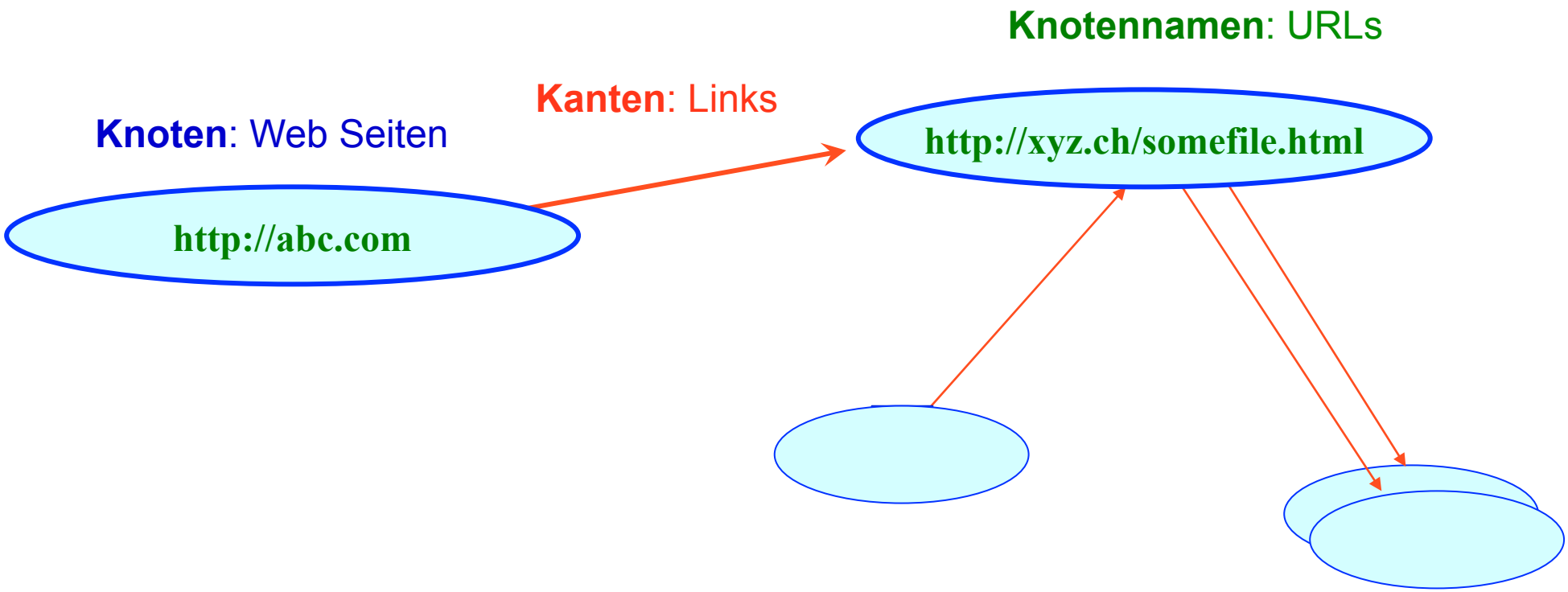
Beispiel: Das World Wide Web

Web Seiten identifiziert durch eindeutige Adressen, sog. **URLs**

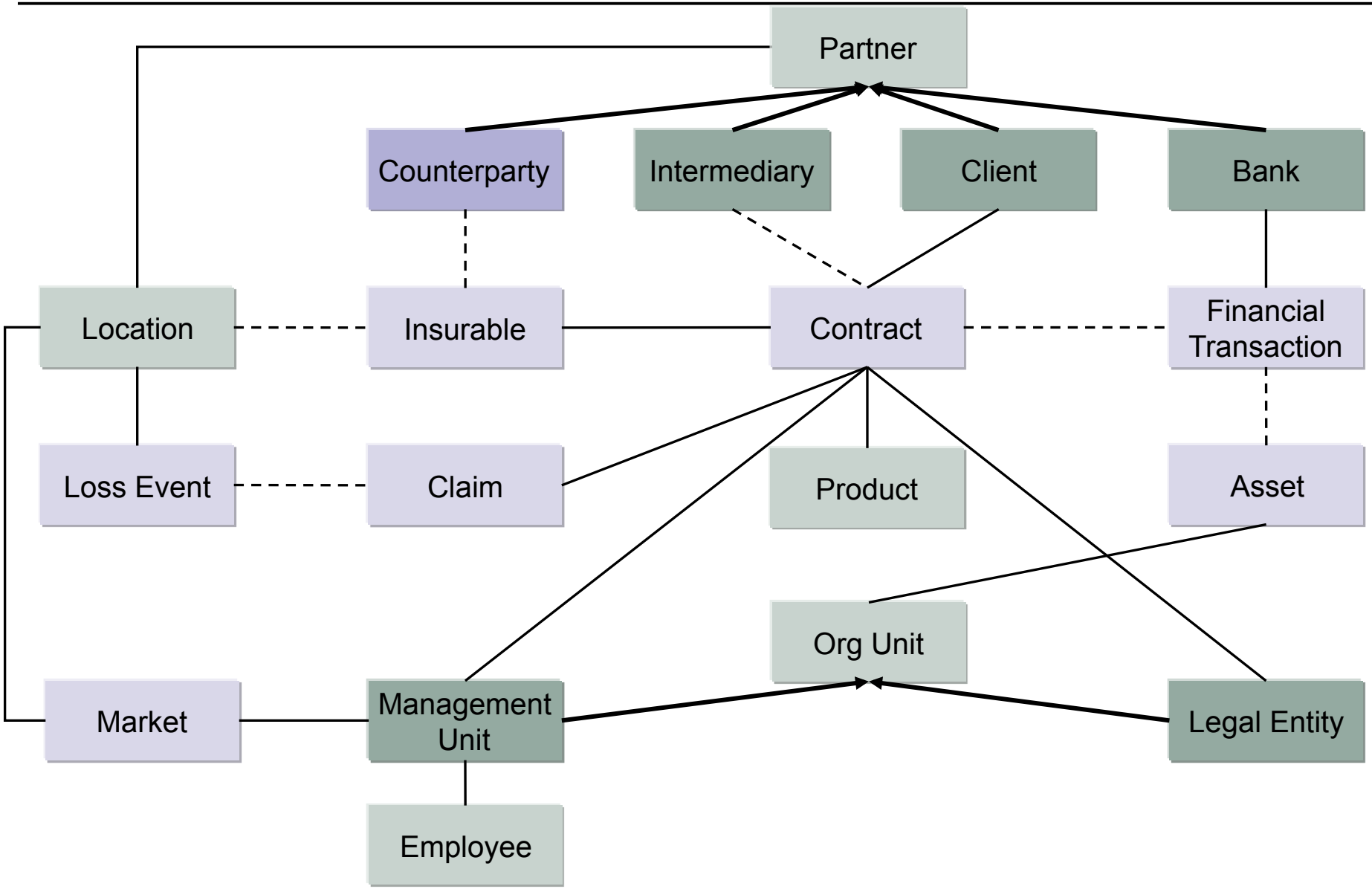


HTML (HyperText Markup Language): Sprache zur Beschreibung von Web Seiten

Beispiel: Das World Wide Web als Graph

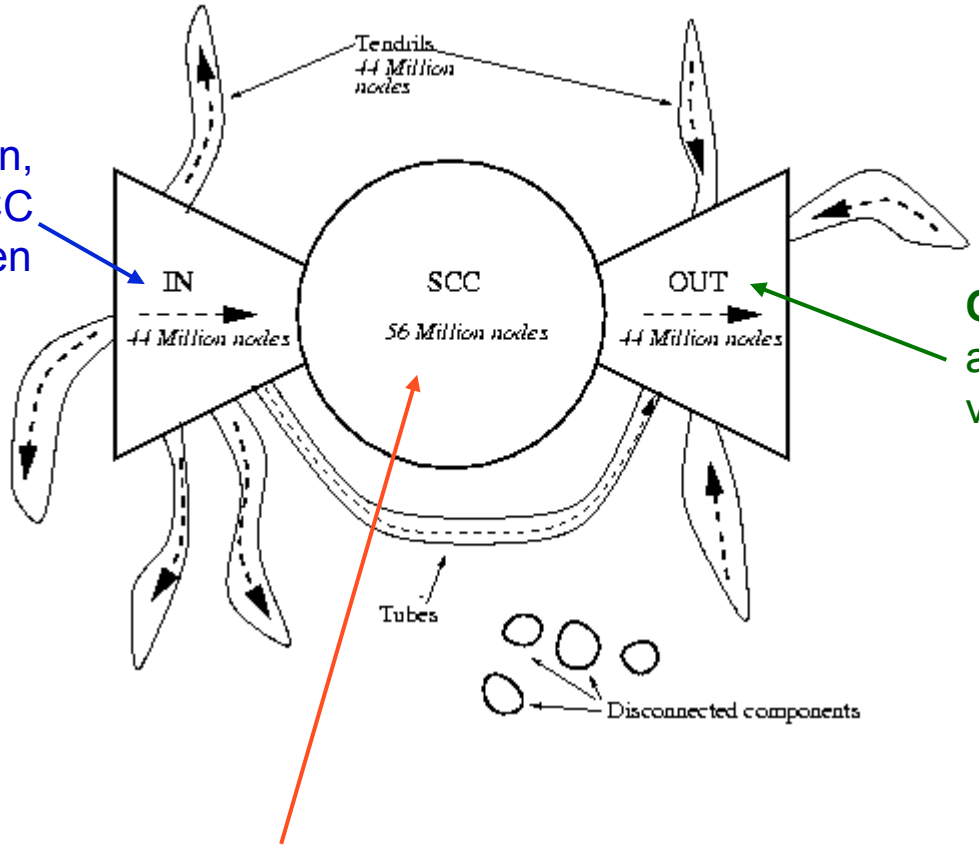


Beispiel: Entity-Relationship Diagramm für Datenbank



Grob-Struktur des Web

gemäss Broder 2000



IN: Menge von Knoten, die auf Knoten im SCC verweisen

OUT: Menge von Knoten, auf die von Knoten im SCC verwiesen wird

SCC: **S**trongly **C**onnected **C**omponent

Ein gerichteter Graph $G = <V, E>$, wobei V die Menge der Knoten (**vertices**) und E die Menge der Kanten (**edges**) ist, heisst **stark zusammenhängend** (strongly connected) falls jeder Knoten $a \in V$ von jedem anderen Knoten $b \in V,\ b \neq a$, erreicht werden kann.

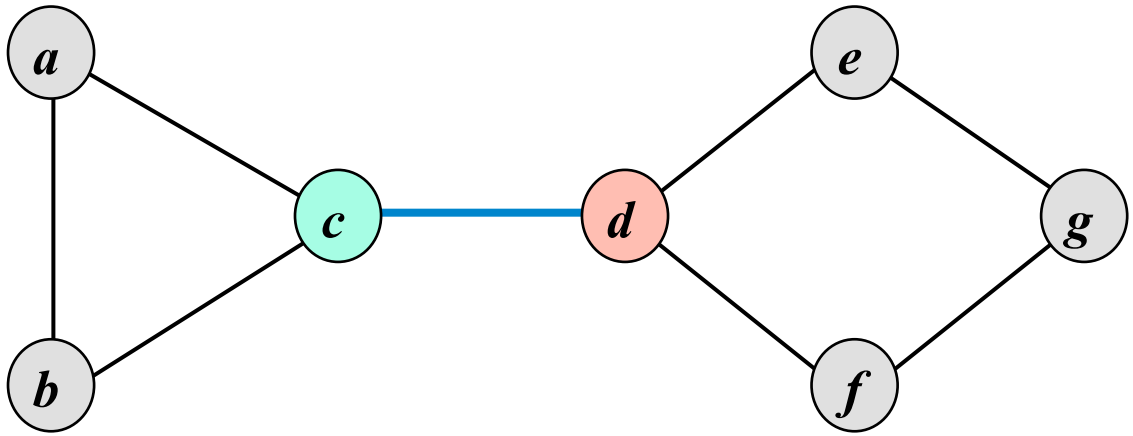
Definition eines Graphs

Ein **ungerichteter Graph** bzw. nur "ein **Graph**" $G = (V, E)$ besteht aus

- eine Menge V von Knoten (*nodes*, ***v**ertices*) und
- einer Menge $E \subseteq V \times V$ von Kanten (***e**dges*) der Form (u,v), welche die Knoten u und v miteinander verbinden: $E = \{ (u,v) \mid u, v \in V \}$

Beispiel: $V = \{ a, b, c, d, e, f, g \}$

$E = \{ (a,b), (a,c), (b,c), (c,d), (d,e), (d,f), (e,g) , (f,g) \}$



Bem.: z.B. $E = \{ (b,a), (a,c), (c,b), (d,c), (e,d), (d,f), (e,g) , (g,f) \}$ ist äquivalent

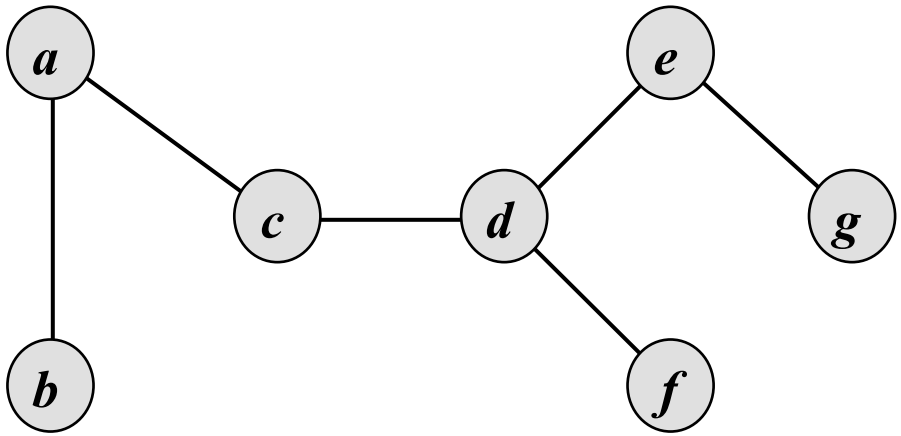
Baum als Spezialfall eines Graphs

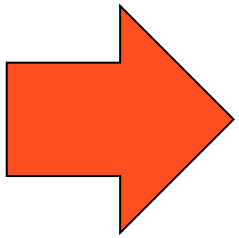
Ein Baum kann als Spezialfall eines Graphen betrachtet werden.
Restriktionen für Bäume:

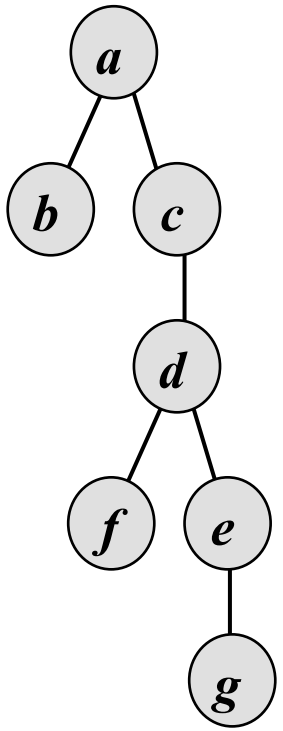
- Jeder Knoten (ausser der Wurzelknoten) hat genau einen "Vater"-Knoten
- Es sind keine Zyklen zugelassen

Beispiel: $V = \{ a, b, c, d, e, f, g \}$

$E = \{ (a,b), (a,c), \sout{(b,c)}, (c,d), (d,e), (d,f), (e,g) , \sout{(f,g)} \}$



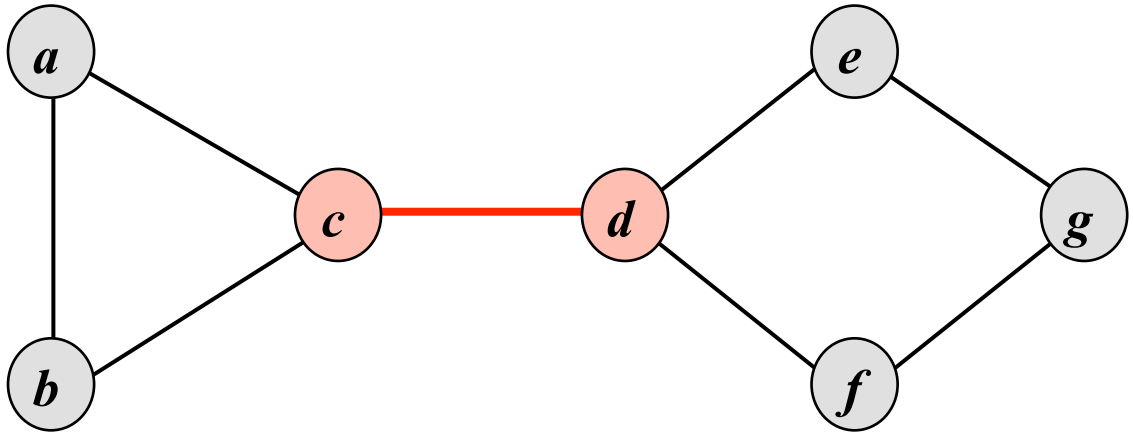




Nachbarschaftsbeziehungen: Adjazenz

Zwei Knoten heissen **adjazent** (adjacent), wenn sie durch eine Kante miteinander verbunden sind.

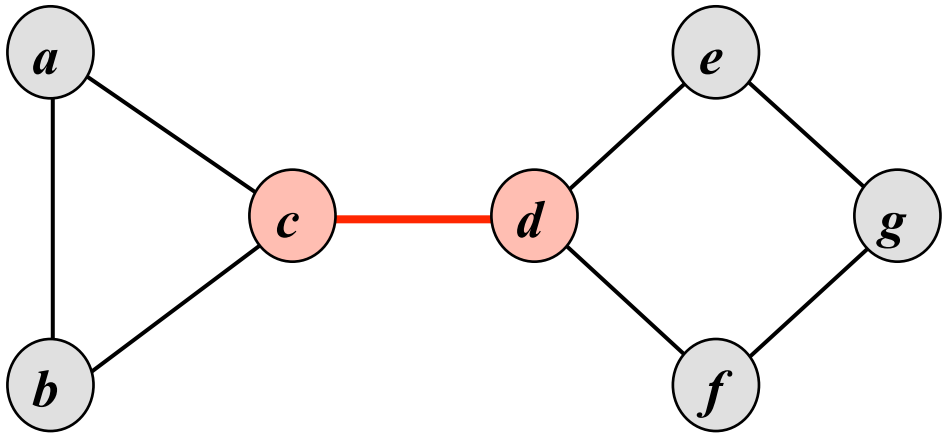
Beispiel: Die Knoten c und d sind adjazent (wie auch z.B. c und a), nicht aber z.B. die Knoten a und d



Adjazenzmatrix

Die Adjazenz zweier Knoten kann in einer Adjazenzmatrix M festgehalten werden, wobei gilt:

$M[x,y] = 1$ falls x und y adjazent, $M[x,y] = 0$ sonst

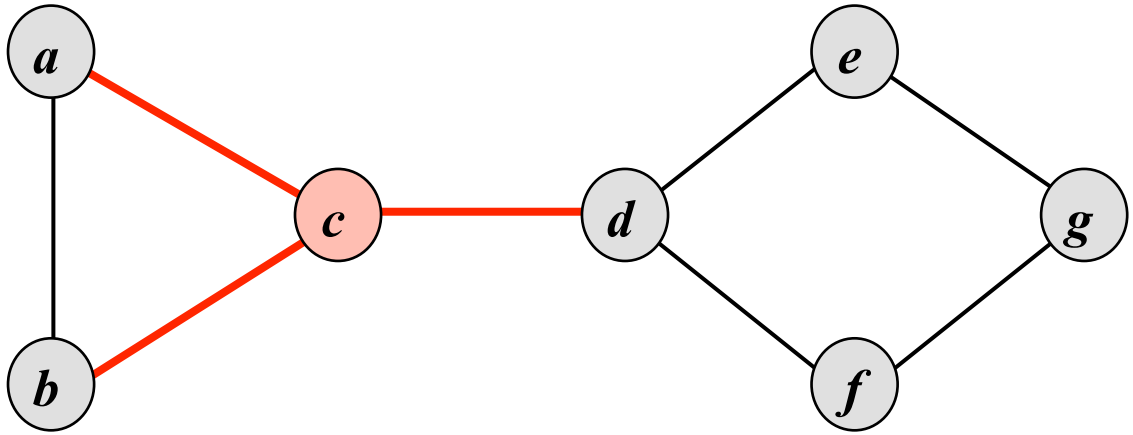


M	a	b	c	d	e	f	g
a	0	1	1	0	0	0	0
b	1	0	1	0	0	0	0
c	1	1	0	1	0	0	0
d	0	0	1	0	1	1	0
e	0	0	0	1	0	0	1
f	0	0	0	1	0	0	1
g	0	0	0	0	1	1	0

Nachbarschaftsbeziehungen: Inzidenz

Ein Knoten heisst **inzident** (incident) zu einer Kante, wenn der Knoten Endpunkt dieser Kante ist.

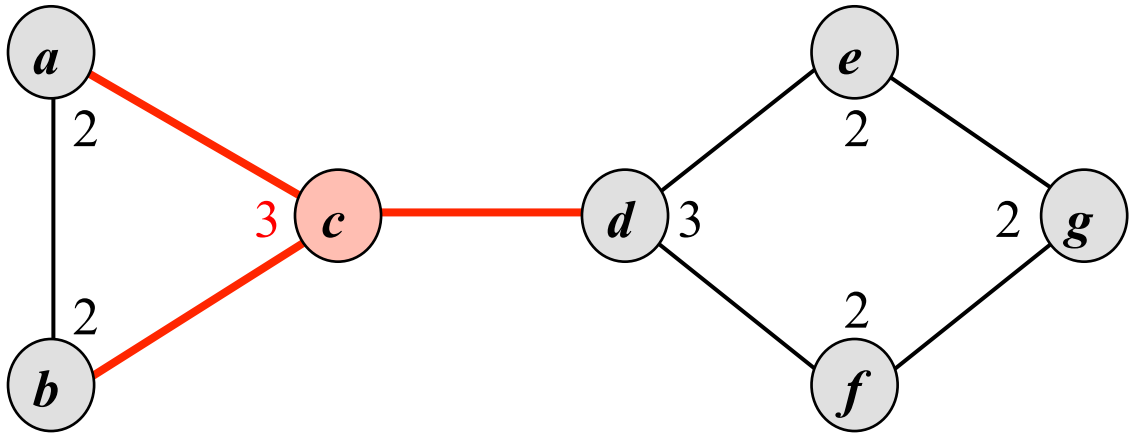
Beispiel: c ist inzident zu den Kanten (a,c), (b,c), und (c,d)



Grad eines Knotens

Die Anzahl der Kanten, für die ein Knoten inzident ist bezeichnet man als **Grad** (*degree*) dieses Knotens.

Beispiel: Der Grad des Knotens c ist 3

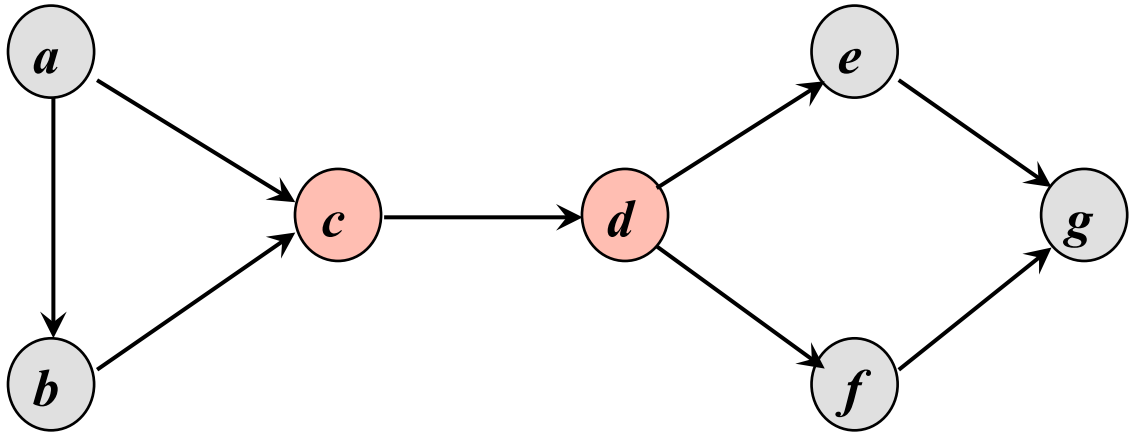


Gerichtete Graphen

Ein Graph heisst **gerichtet** (*directed*), wenn seine Kanten eine Orientierung (von einem "Start"-Knoten zu einem "End"-Knoten) aufweisen. Ein gerichteter Graph wird auch als **Digraph** (***di****rected* ***graph***) bezeichnet.

Beispiel:

$$V = \{ a, b, c, d, e, f, g \}$$
$$E = \{ (a,b), (a,c), (b,c), (c,d), (d,e), (d,f), (e,g) , (f,g) \}$$

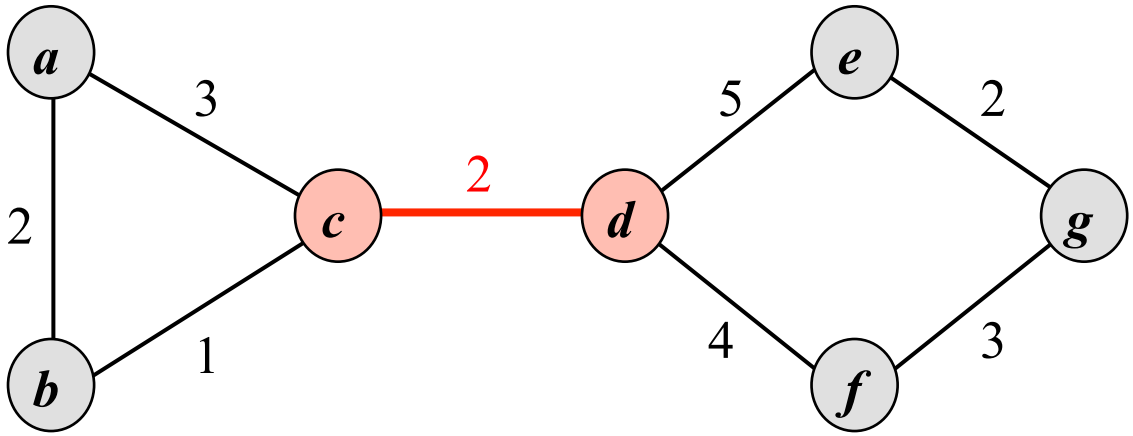


Gewichtete Graphen

Ein Graph heisst **gewichtet** (*weighted*), wenn seinen Kanten Attribute (Gewichte) zugeordnet sind.

zB: Entfernungen in einem Ortsnetz

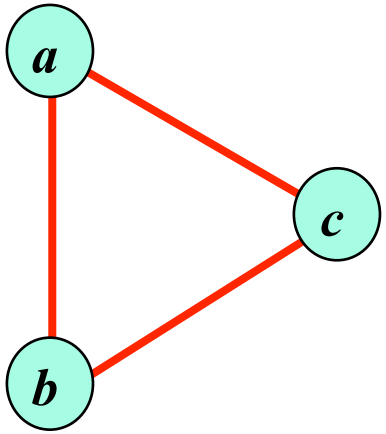
Beispiel: Die Kante (c,d) hat das Gewicht 2.

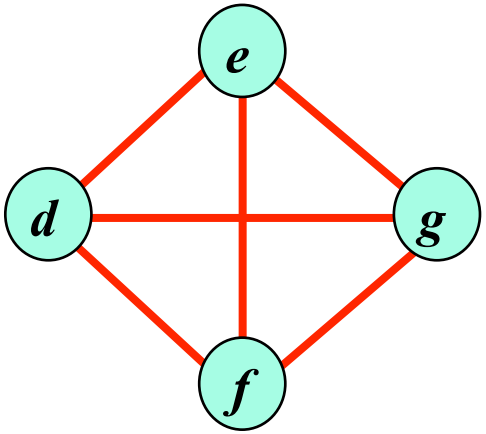


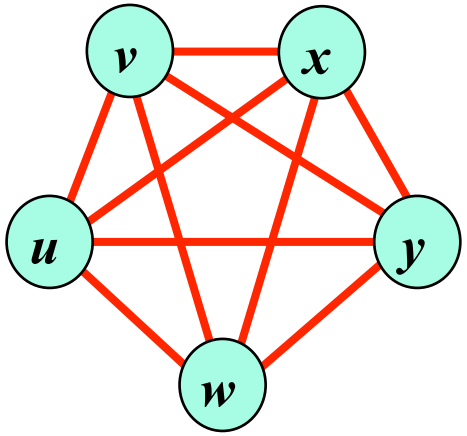
Vollständige Graphen

Ein Graph heisst **vollständig** (*complete*), wenn eine Kante von jedem Knoten zu jedem anderen Knoten führt.

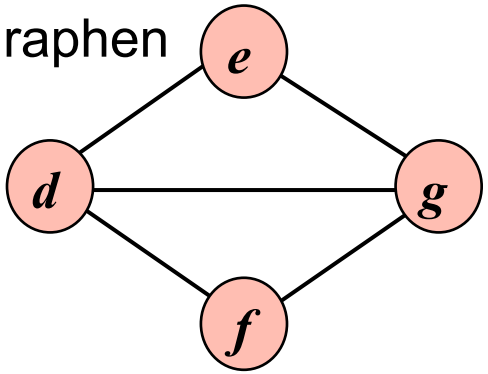
Beispiele **vollständiger** Graphen: n Knoten $\Rightarrow$ $n \cdot (n-1) \,/\, 2$ Kanten

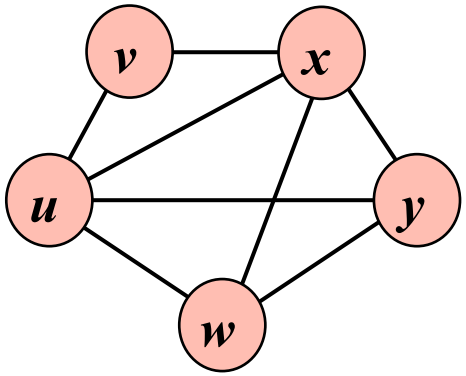






Beispiele unvollständiger Graphen

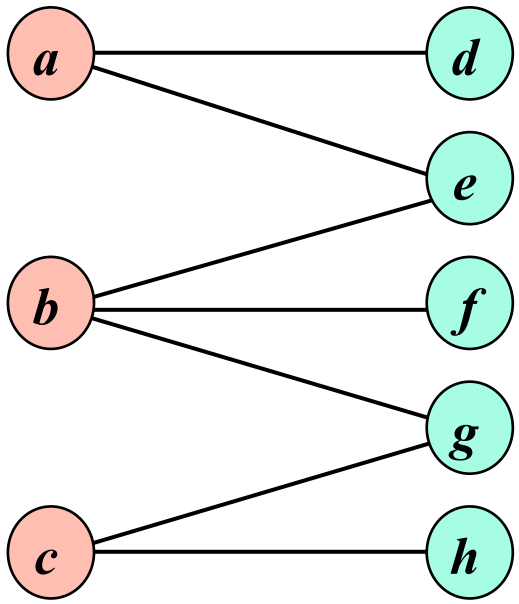




Bipartite Graphen

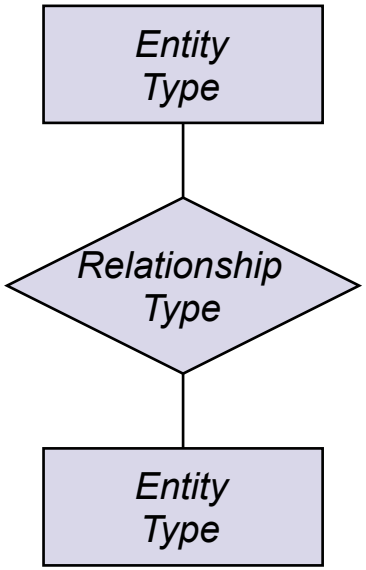
Ein Graph heisst **bipartit** (*bipartite*), wenn Kanten nur Knoten aus zwei disjunkten Teilmengen miteinander verbinden.

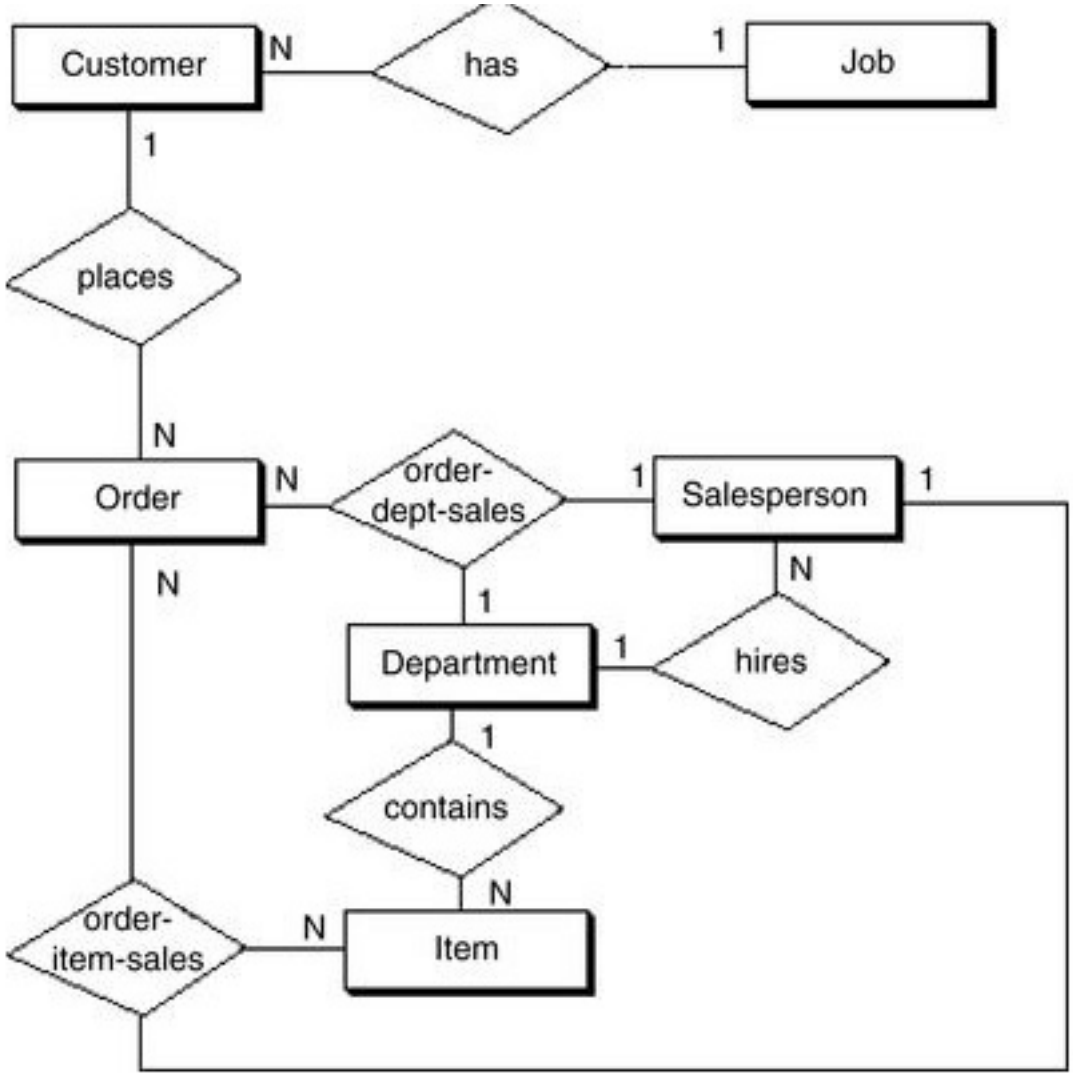
zB:



Beispiel: "Klassisches" ER-Diagramm als bipartiter Graph

Ein Knoten gehört entweder zur Teilmenge der "Entity Types" (Objekttypen) oder zur Teilmenge der "Relationship Types" (Beziehungstypen).





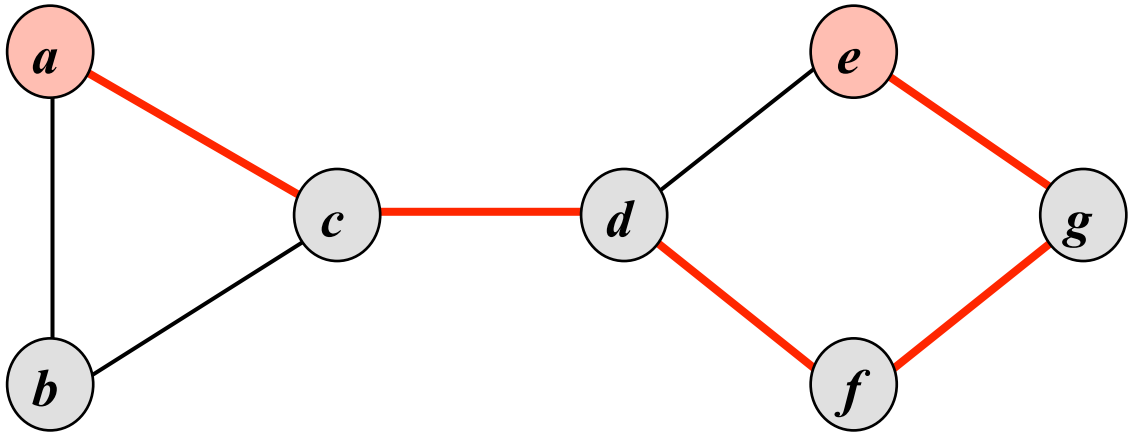
Pfade

Eine Folge von n Kanten, die zwei Knoten u und v miteinander verbindet, wird als Weg oder **Pfad** (*path*) der Länge n bezeichnet.

Formal: Eine Folge von Knoten $x_1, x_2, ..., x_n \in V$ mit $x_1 := u$ und $x_n := v$ in einem Graphen $G = (V, E)$, $E \subseteq V \times V$ heisst **Pfad** der Länge n $(n \geq 1)$ sofern gilt:

$$\forall\, i:\ 1 \leq i < n :\ (x_i, x_{i+1}) \in E$$

Pfad der Länge $n = 5$ von a nach e: [$(a,c), (c,d), (d,f), (f,g), (g,e)$]
bzw. [a, c, d, f, g, e]



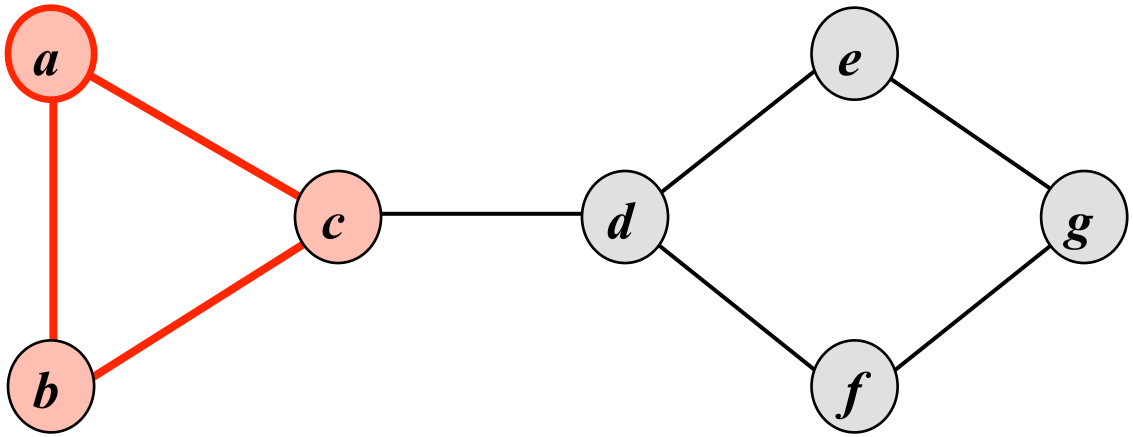
Zyklen

In einem ungerichteten Graphen heisst ein geschlossener Pfad, der zu seinem Startknoten zurückführt und eine Länge $n \geq 3$ hat, **Zyklus** (*cycle*).

Formal: Eine Folge von Knoten $x_1, x_2, ..., x_n \in V$ mit $x_1 := u$ und $x_n = x_1 = u$ sowie $n \geq 3$ in einem ungerichteten Graphen $G = (V, E)$, $E \subseteq V \times V$ heisst **Zyklus** der Länge n sofern gilt:

$$\forall\, i:\ 1 \leq i < n :\ (x_i, x_{i+1}) \in E$$

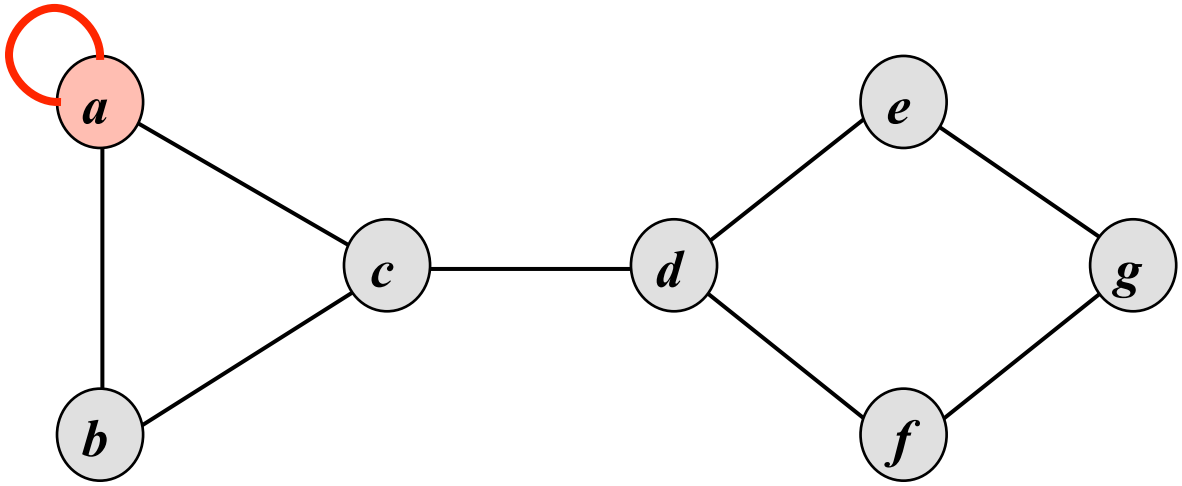
Beispiel: Zyklus der Länge $n = 3$ von a nach a: a, b, c, a



Schlaufen

Eine Kante, die einen Knoten mit sich selbst verbindet heisst **Schlaufe** (*loop*).

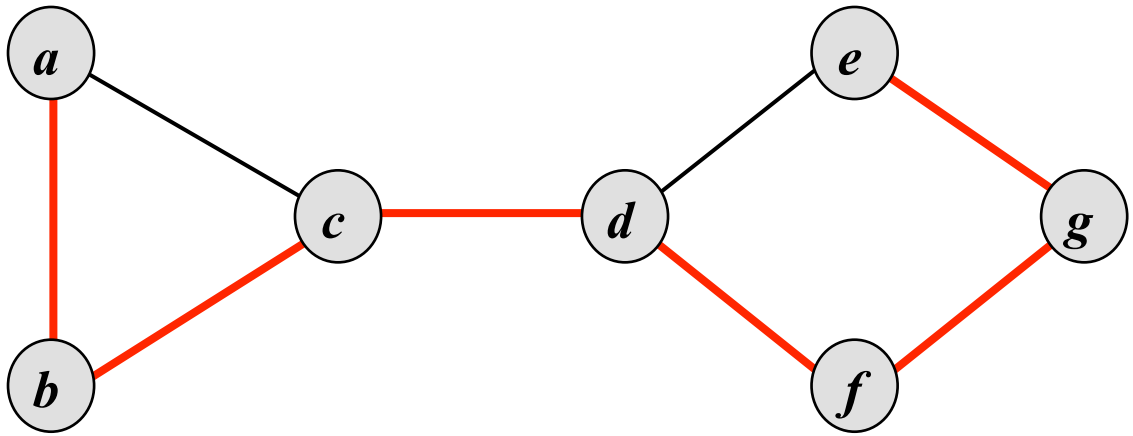
Beispiel: $E = \{ (a,a), (a,b), (a,c), (b,c), (c,d), (d,e), (d,f), (e,g) , (f,g) \}$



Hamilton Pfad

In einem **Hamilton Pfad** wird jeder Knoten des Graphen genau einmal durchlaufen.

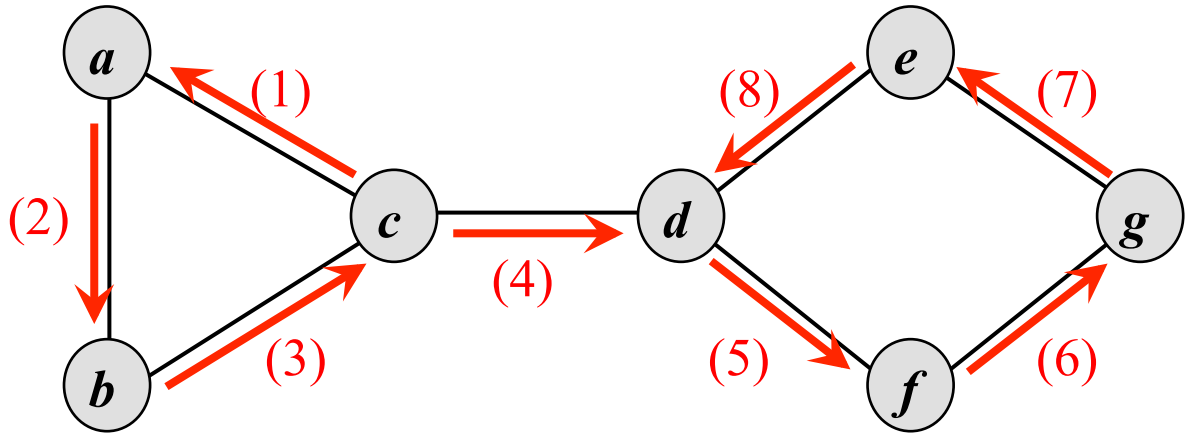
Beispiel:



Euler Pfad

In einem **Euler Pfad** ist jede Kante des Graphen genau einmal enthalten.

Beispiel
(Die Pfeile geben an, wie die ungerichteten Kanten durchlaufen werden.)



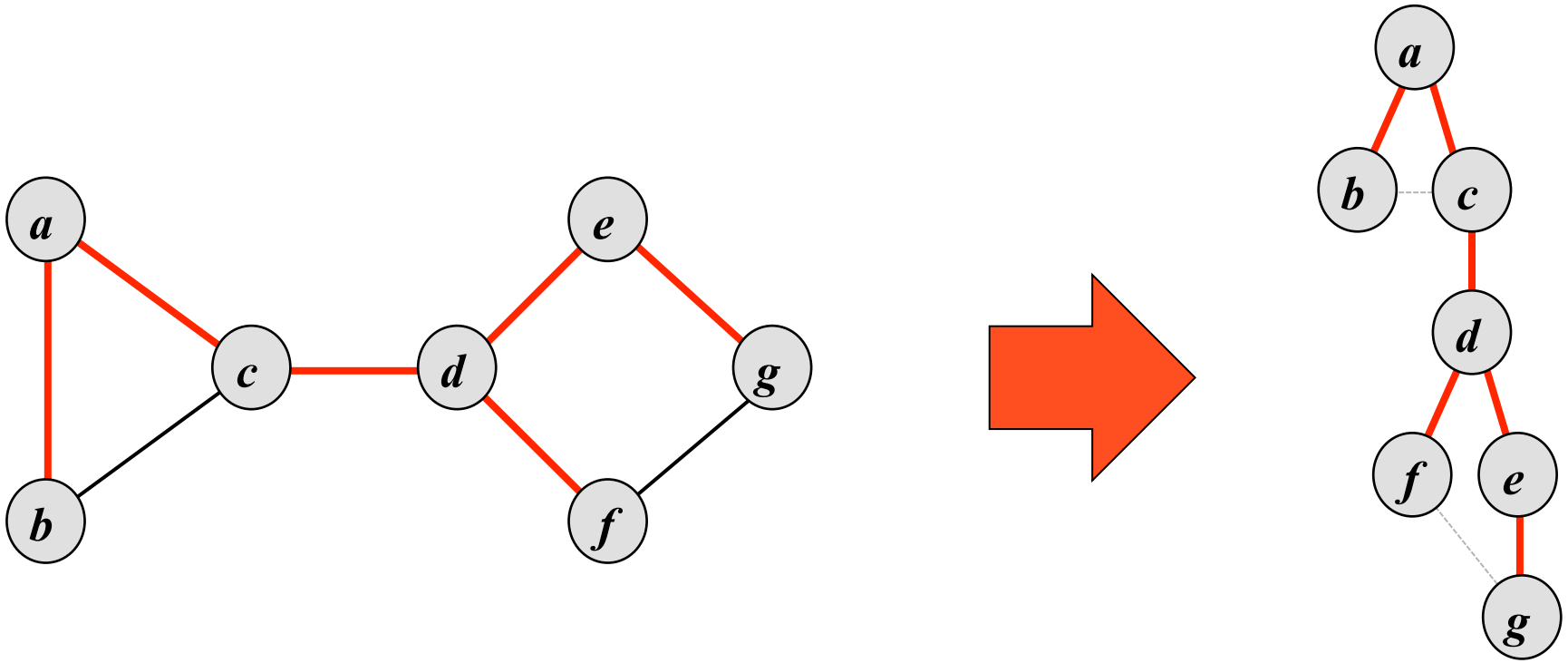
Spannender Baum

Ein **spannender Baum** (*spanning tree*) S eines Graphen verbindet alle Knoten durch eine zyklenfreie Teilmenge aller Kanten E.

Beispiel: $S = \{ (a,b), (a,c), (c,d), (d,e), (d,f), (e,g) \}$

$E = \{ (a,b), (a,c), (b,c), (c,d), (d,e), (d,f), (e,g) , (f,g) \}$

$S \subseteq E$



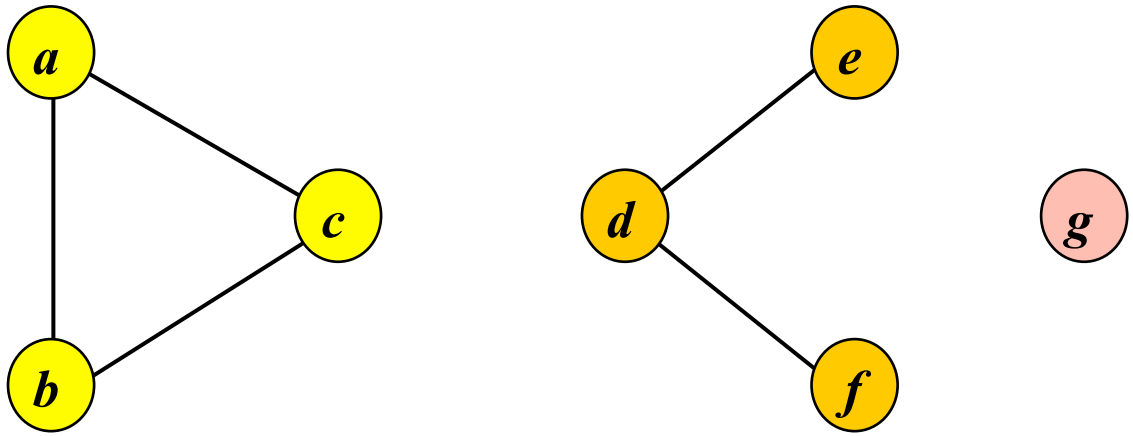
Komponenten

Eine **Komponente** (*component*) ist ein zusammenhängender Teil eines Graphen.

Ein Graph kann aus mehreren Komponenten bestehen.

Beispiel: Graph bestehend aus 3 Komponenten $\{ a, b, c \}, \{ d, e, f \}, \{ g \}$

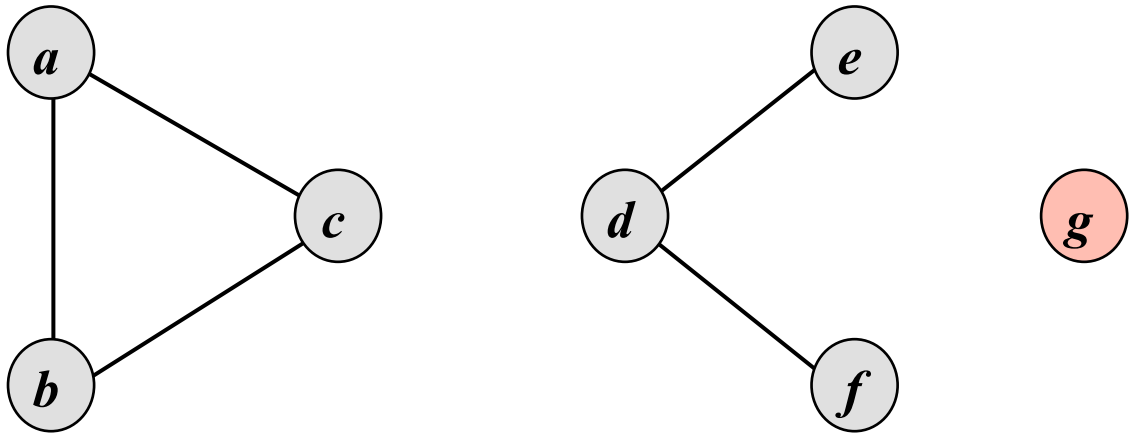
$E = \{ (a,b), (a,c), (b,c), (d,e), (d,f) \}$



Isolierter Knoten

Ein Knoten heisst **isoliert** (*isolated*), wenn er der einzige Knoten einer Komponente ist.

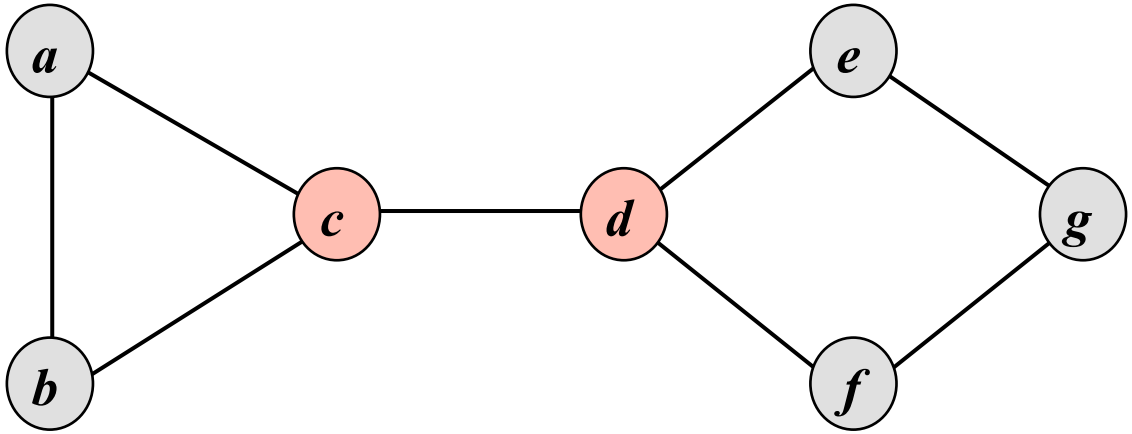
Beispiel: Graph mit einem isolierten Knoten



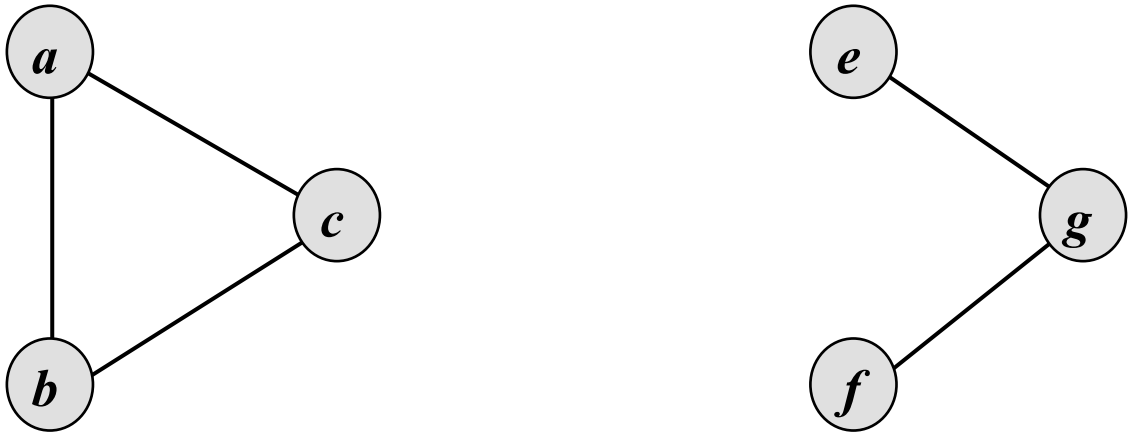
Kritische Knoten

Ein **Knoten** eines Graphen ist genau dann **kritisch** (*critical*), wenn durch seine Entfernung der Graph in mehrere Komponenten zerfällt.

Beispiel: Graph mit kritischen Knoten c und d :



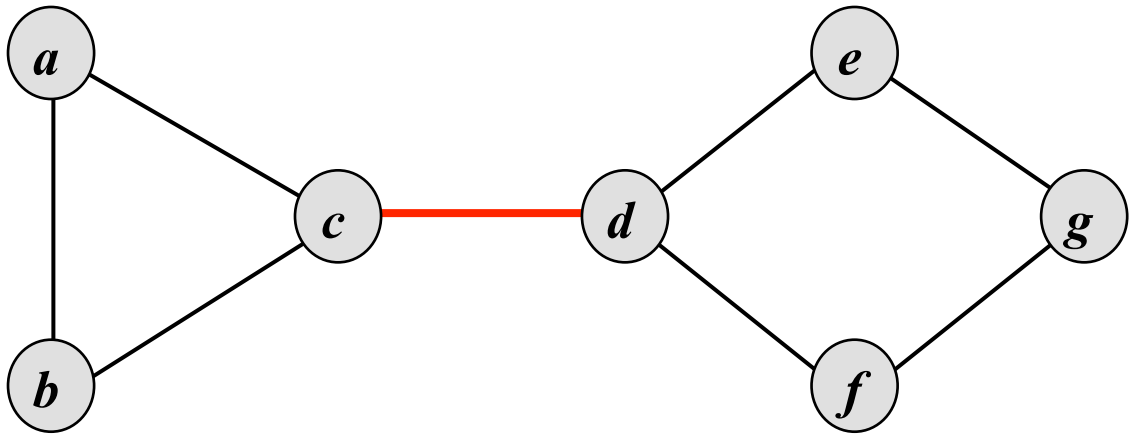
Graph ohne Knoten d:



Kritische Kanten

Eine **Kante** eines Graphen ist genau dann **kritisch** (*critical*), wenn durch ihre Entfernung der Graph in mehrere Komponenten zerfällt.

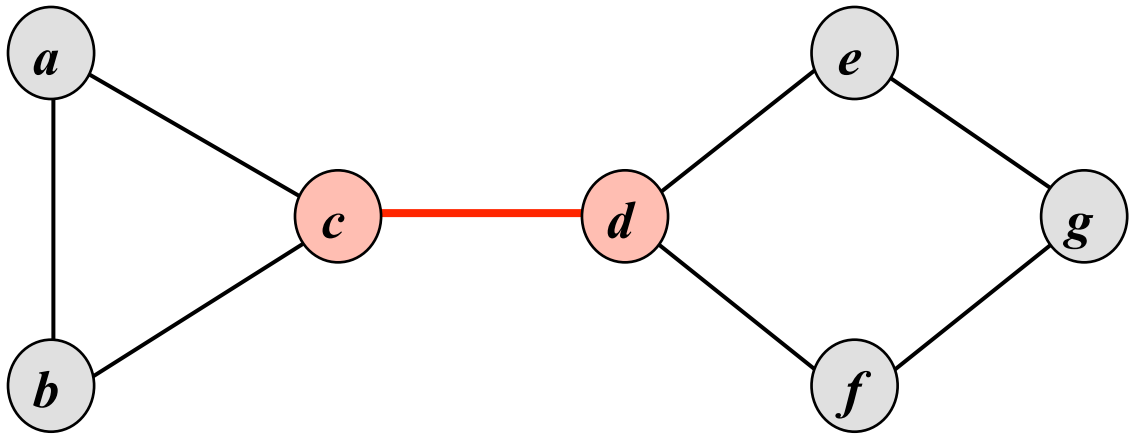
Beispiel: Kritische Kante (c,d)



Artikulationspunkte

Kritische Kanten und kritische Knoten werden gemeinsam auch als **Artikulationspunkte** (*articulation points*) eines Graphen bezeichnet.

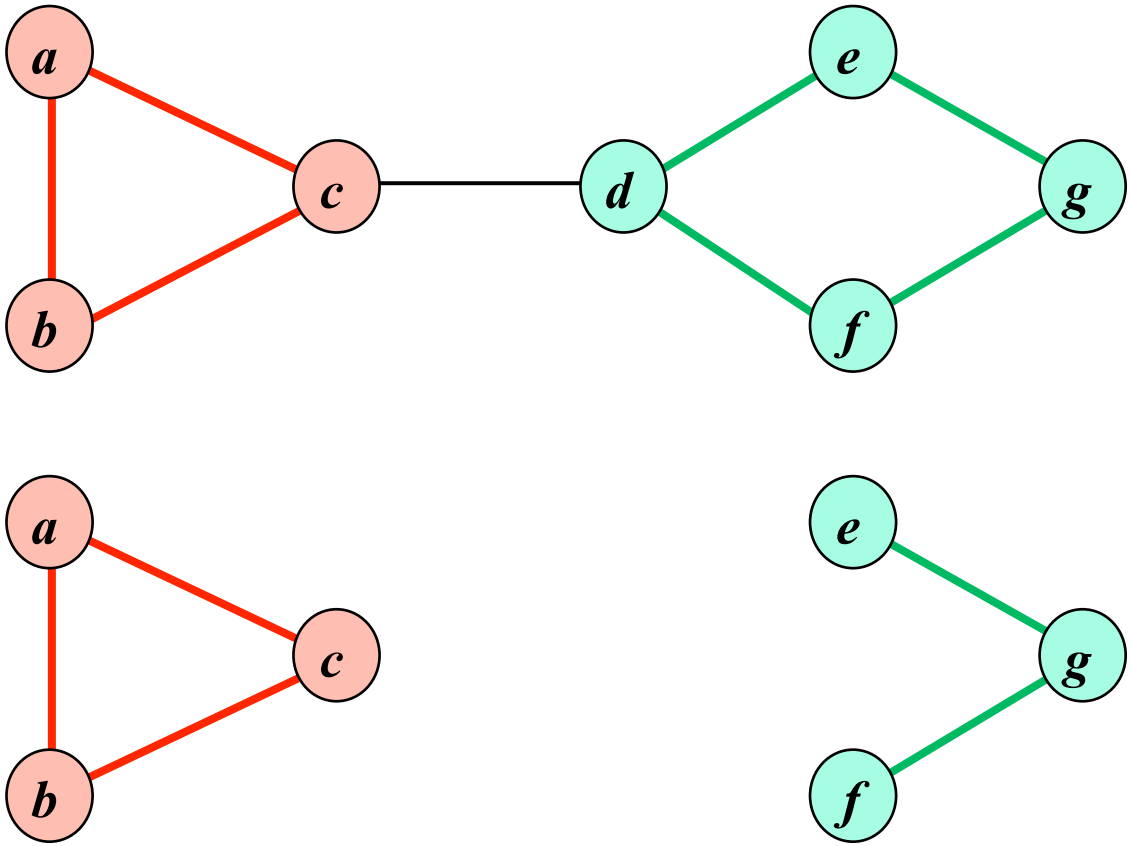
Beispiel: Artikulationspunkte: Knoten c und d, Kante (c,d)



Zweifacher Zusammenhang

Eine Komponente eines ungerichteten Graphen ist **zweifach zusammenhängend** (*biconnected*), wenn nach Entfernen eines beliebigen Knotens die verbleibenden Knoten zusammenhängend sind.

Beispiel:
Entfernen des Knotens d:



Schwacher Zusammenhang

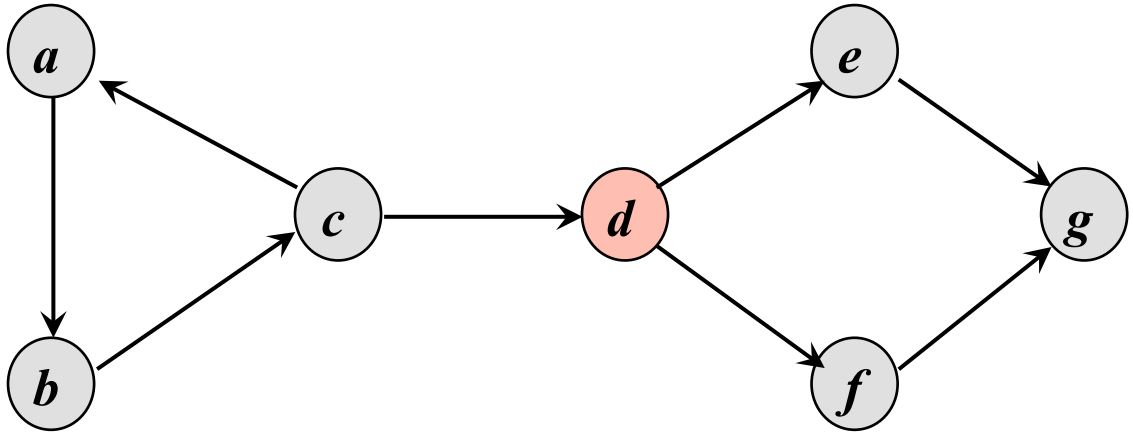
In einem Digraphen (= in einem gerichteten Graphen) heisst ein Knoten y von einem Knoten x aus **schwach erreichbar** (*weakly reachable*), wenn es einen ungerichteten Pfad von x nach y gibt.

Eine **Komponente** eines Digraphen ist **schwach zusammenhängend** (*weakly connected*), wenn jeder Knoten von jedem anderen Knoten aus schwach erreichbar ist.

Beispiel: Alle Knoten sind von d aus schwach erreichbar, z.B.

g "vorwärts" via Pfad $[\,(d,e), (e,g)\,]$

a "rückwärts" und "vorwärts" via Pfad $[\,(c,d), (c,a)\,]$

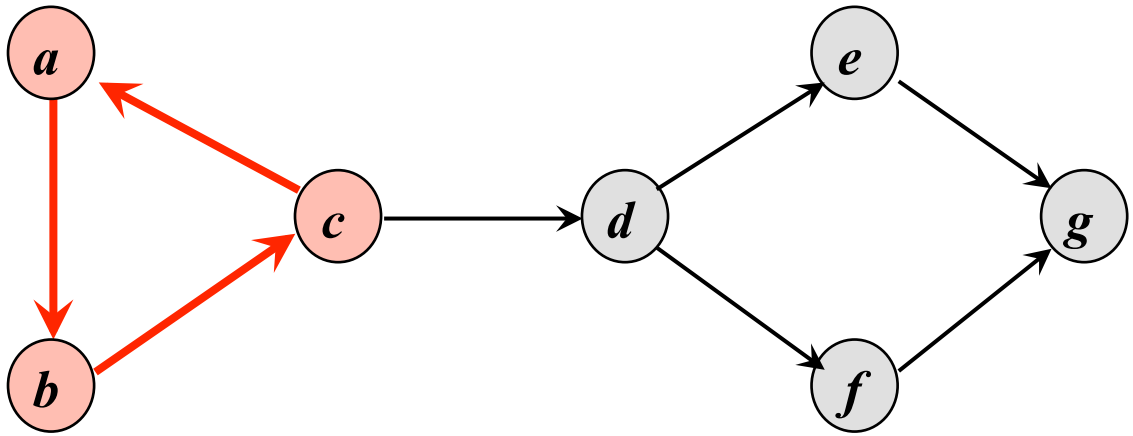


Starker Zusammenhang

In einem Digraphen (= in einem gerichteten Graphen) heisst ein Knoten y von einem Knoten x aus **stark erreichbar**, wenn es einen (gerichteten) Pfad von x nach y gibt.

Eine Komponente eines Digraphen heisst **stark zusammenhängend** (*strongly connected*), wenn jeder Knoten von jedem anderen Knoten aus stark erreichbar ist.

Beispiel einer stark zusammenhängenden Komponente eines Digraphen:

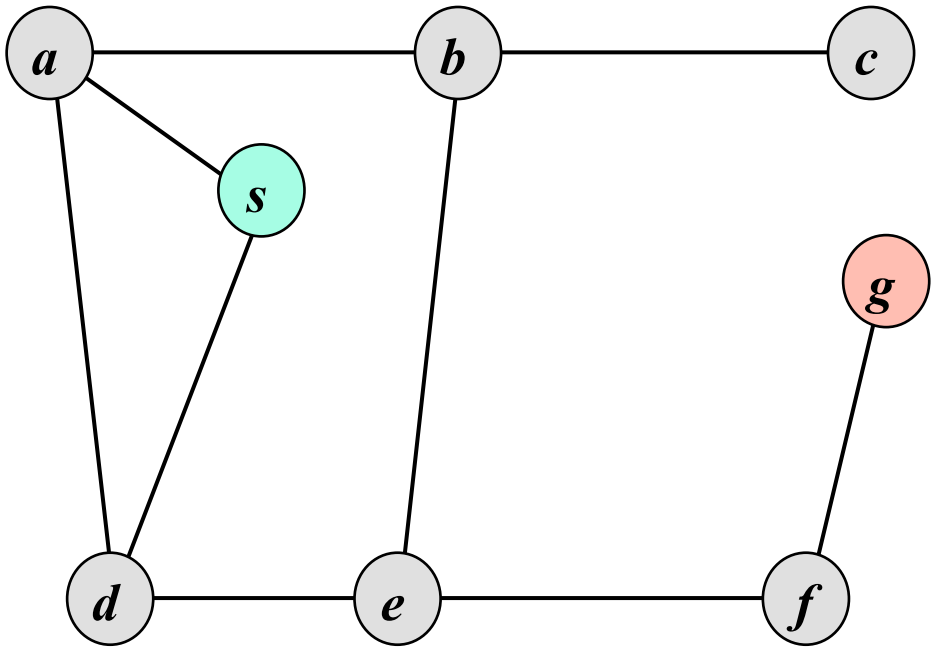


Anwendung: Suche (1)

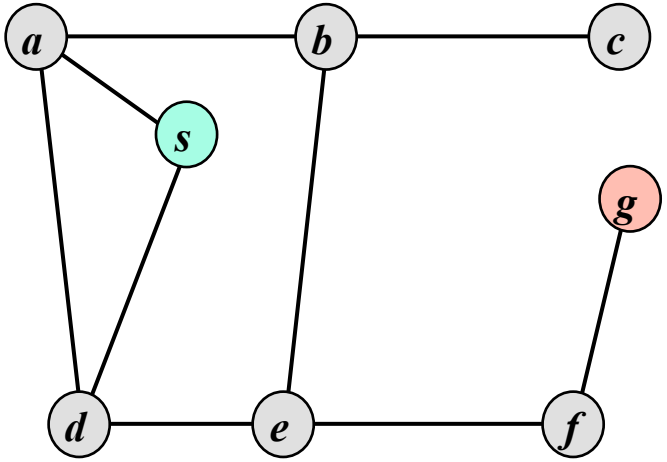
Beispiel eines Verkehrsnetzes

Gibt es einen Pfad von Start *s* zum Ziel (goal) *g*?

Welchen?



Anwendung: Suche (2)

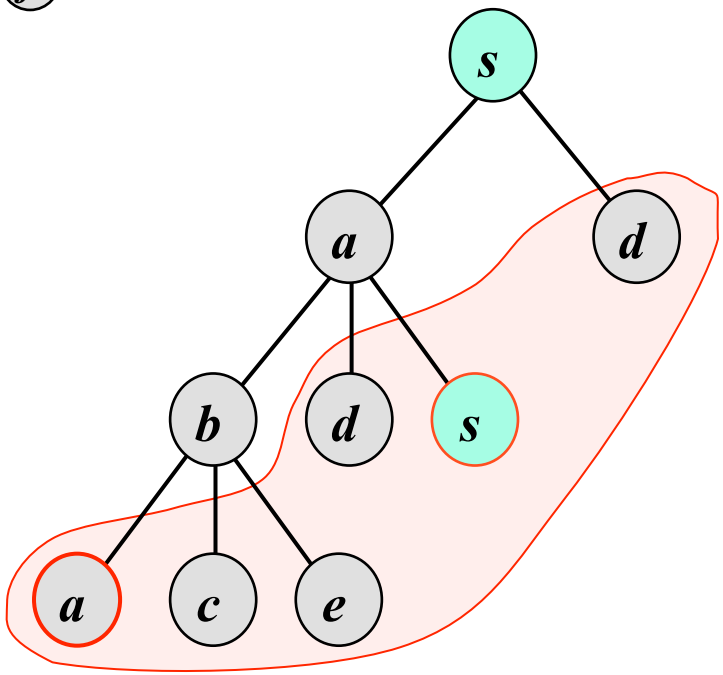


Grundprinzip: Konstruktion eines Suchbaums und Führen einer Liste offener Knoten (sog. Agenda), welche die noch zu untersuchenden ("zu expandierenden") Knoten enthält

Es gibt verschiedene Strategien, welcher offene Knoten zuerst expandiert wird.

Jeweils 1. Knoten in Agenda
⇒ **Tiefensuche**

Suchbaum



Agenda

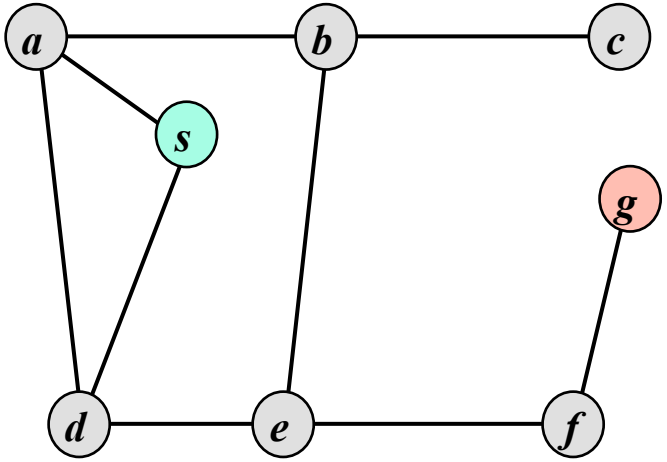
[*s*]

[*a, d*]

[*b, d, s, d*]

[*a, c, e, d, s, d*]

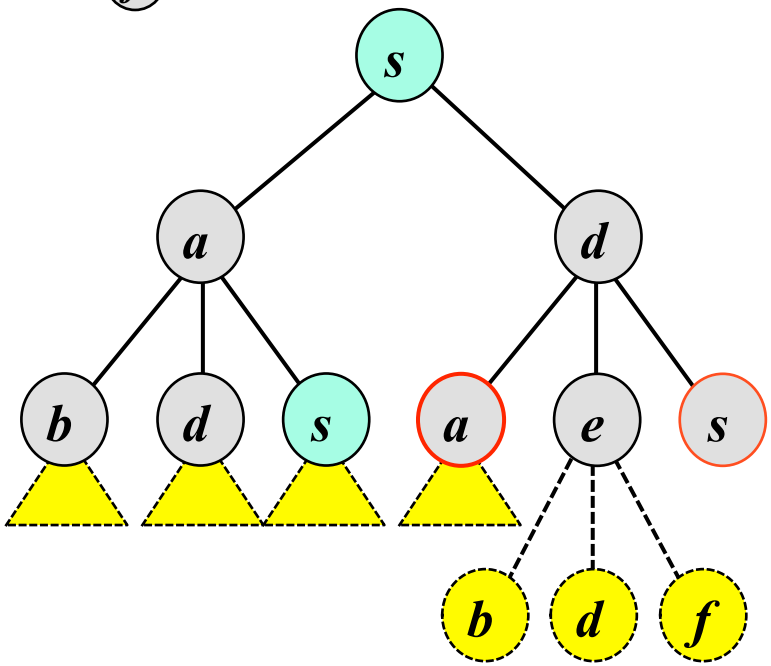
Anwendung: Suche (3)



Breitensuche: Zuerst werden "ältere" Knoten der Agenda expandiert

Suchbaum

Agenda



[*s*]

[*a, d*]

[*b, d, s, d*]

[*b, d, s, a, e, s*]

[… , *b, d, f*]

Analyse Tiefensuche (Platzbedarf)

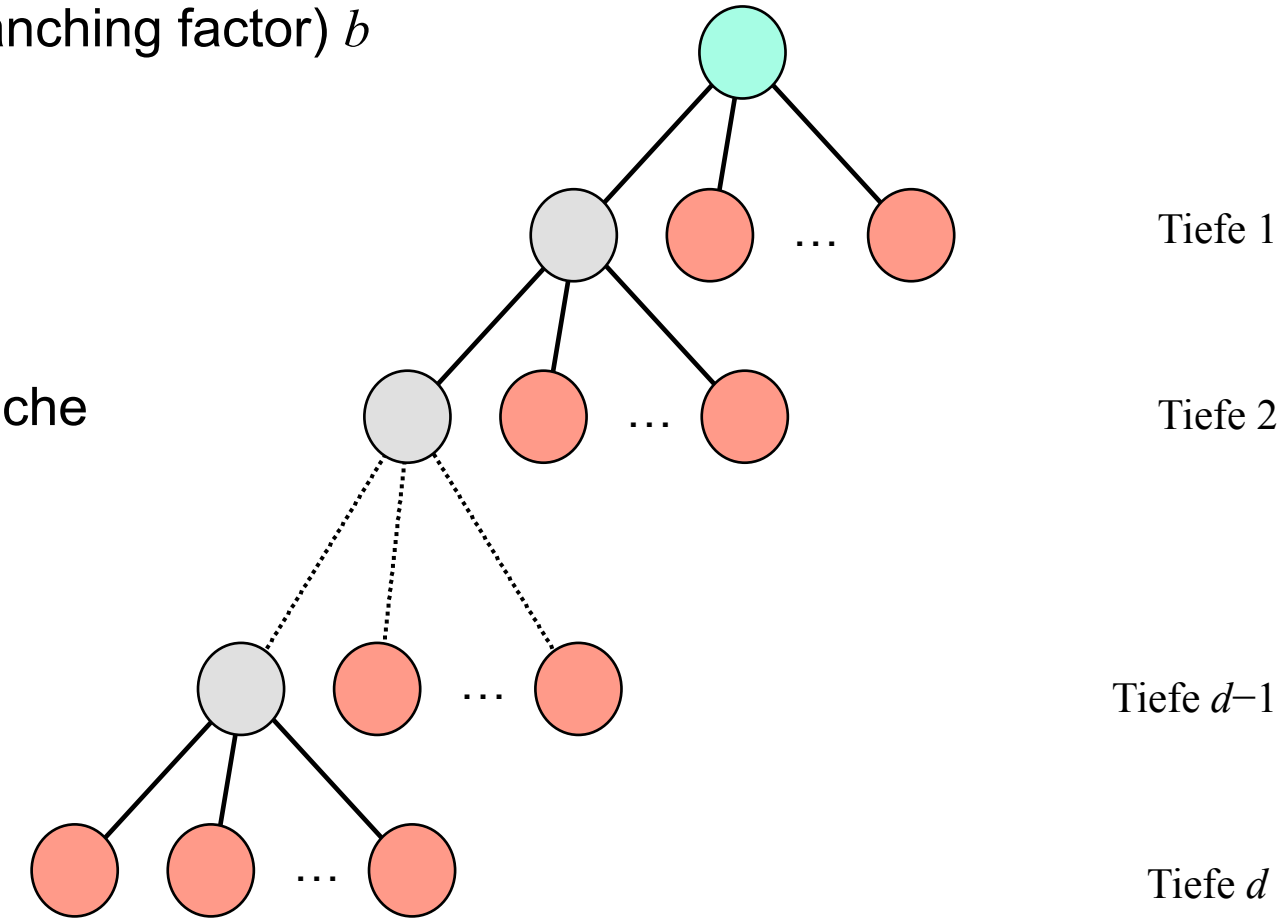
Annahmen:

- mittlerer fanout (branching factor) b
- Ziel in Tiefe d
- keine Zyklen

Platzbedarf bei Tiefensuche
Anzahl offener Knoten

$$(b-1) \cdot (d-1) + b$$
$$= (b-1) \cdot d + 1$$

lineare Funktion
der Tiefe d



Analyse Breitensuche (Platzbedarf)

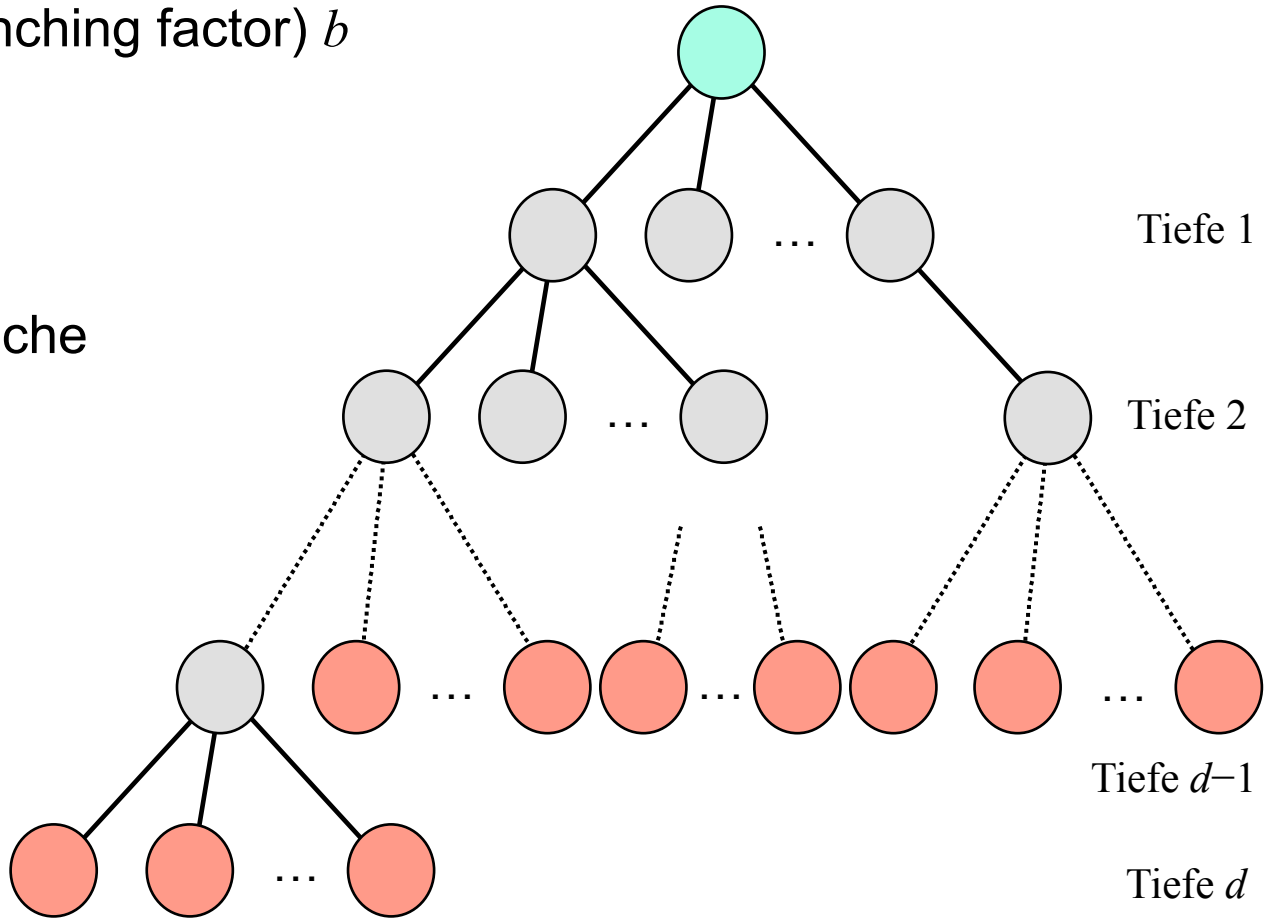
Annahmen:

- mittlerer fanout (branching factor) b
- Ziel in Tiefe d

Platzbedarf bei Breitensuche
Anzahl offener Knoten

$b^{d-1} - 1 + b$

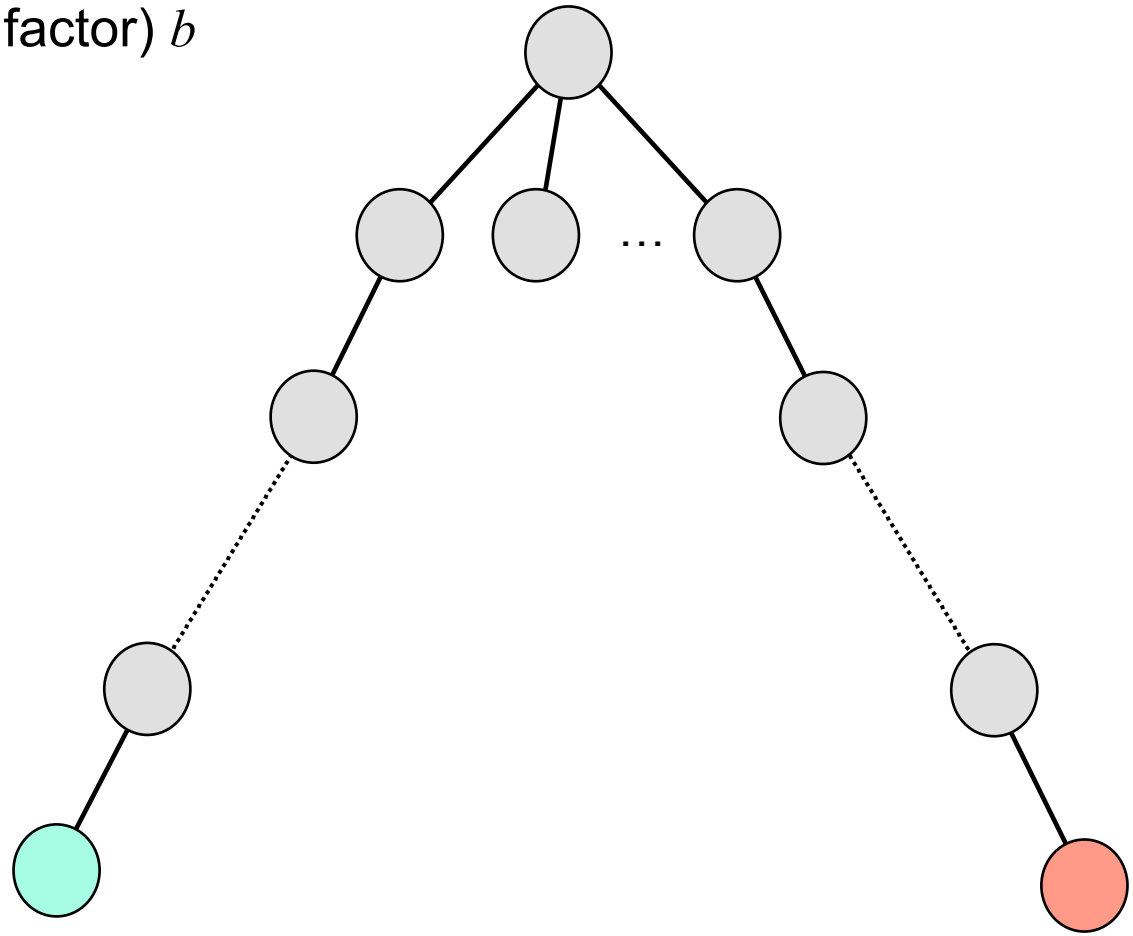
exponentielle Funktion
der Tiefe d



Analyse Tiefensuche (Zeitbedarf)

Annahmen:

- mittlerer fanout (branching factor) b
- Ziel in Tiefe d
- keine Zyklen



Zeitbedarf bei Tiefensuche
Anzahl expandierter Knoten

best: $d + 1$

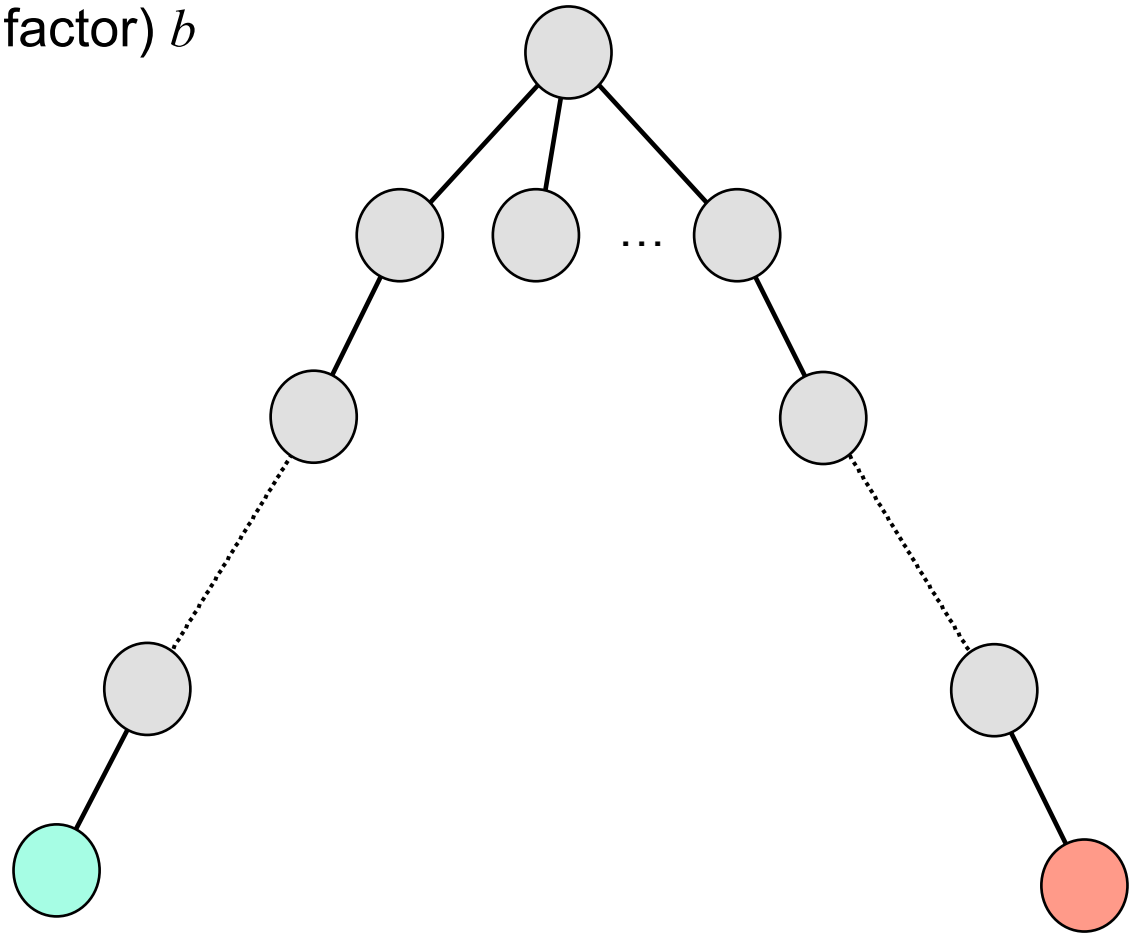
worst: $1 + b + \ldots + b^d$
$= (b^{d+1} - 1) / (b-1)$

exponentielle Funktion
der Tiefe d

Analyse Breitensuche (Zeitbedarf)

Annahmen:

- mittlerer fanout (branching factor) b
- Ziel in Tiefe d



Zeitbedarf bei Breitensuche
Anzahl expandierter Knoten

best: $1 + b + \ldots + b^{d-1} + 1$
$= (b^d - 1) / (b-1) + 1$

worst: $1 + b + \ldots + b^d$
$= (b^{d+1} - 1) / (b-1)$

exponentielle Funktion
der Tiefe d