Einführung zu Graphen

- Ungerichtete Graphen
- Darstellung
- Tiefensuche
  - Zusammenhangskomponenten
- Breitensuche
  - Bipartite Graphen

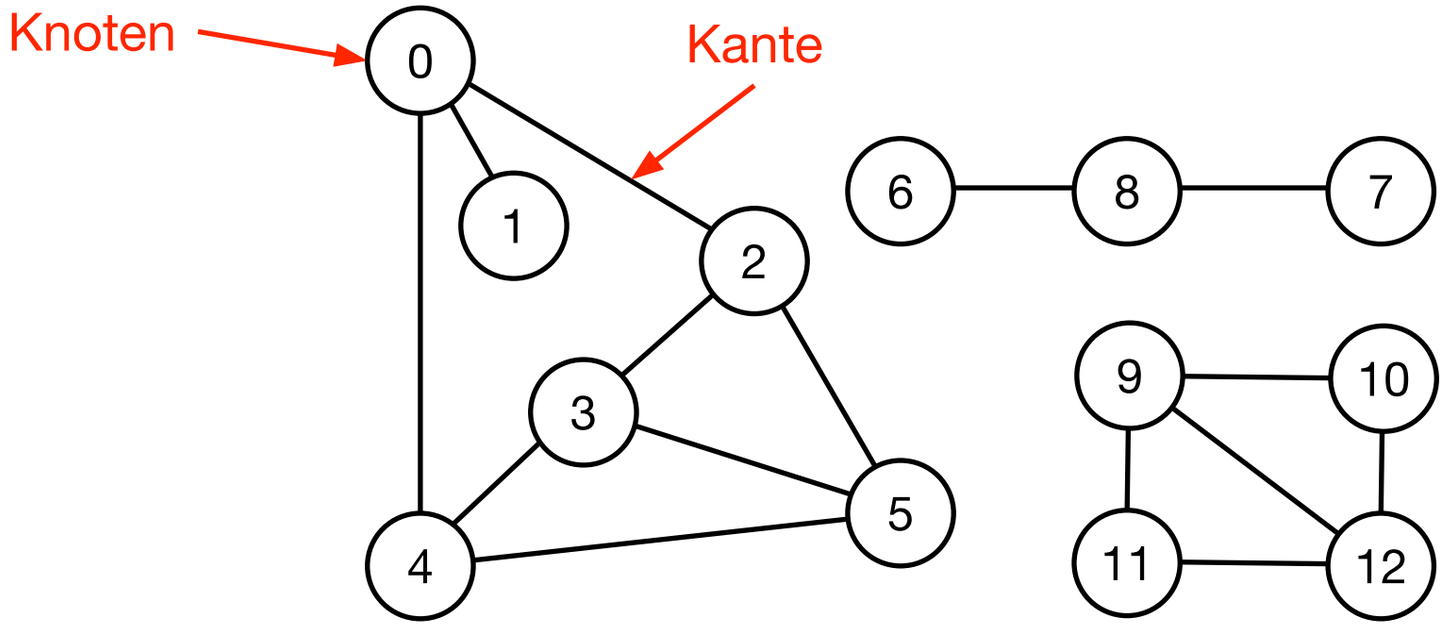
Holger Dell
Folien adaptiert von Philip Bille

Einführung zu Graphen

- Ungerichtete Graphen
- Darstellung
- Tiefensuche
    - Zusammenhangskomponenten
- Breitensuche
    - Bipartite Graphen

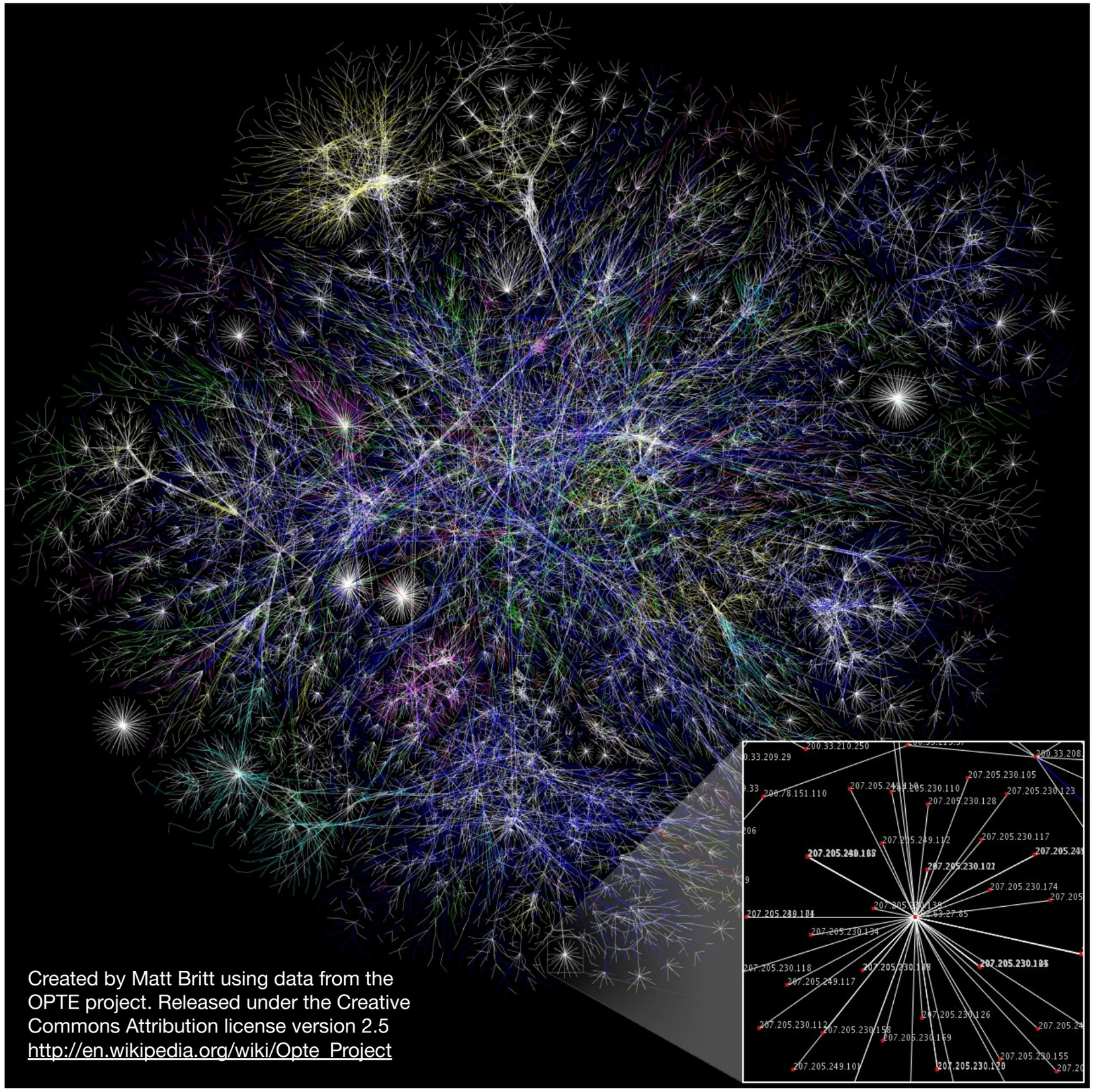
Ungerichtete Graphen

- Ungerichteter Graph. Menge von Knoten, die durch Kanten verbunden sind.



- Wozu Graphen?
  - Modelliert viele natürliche Probleme.
  - Tausende praktische Anwendungen.
  - Hunderte bekannte Graphalgorithmen.

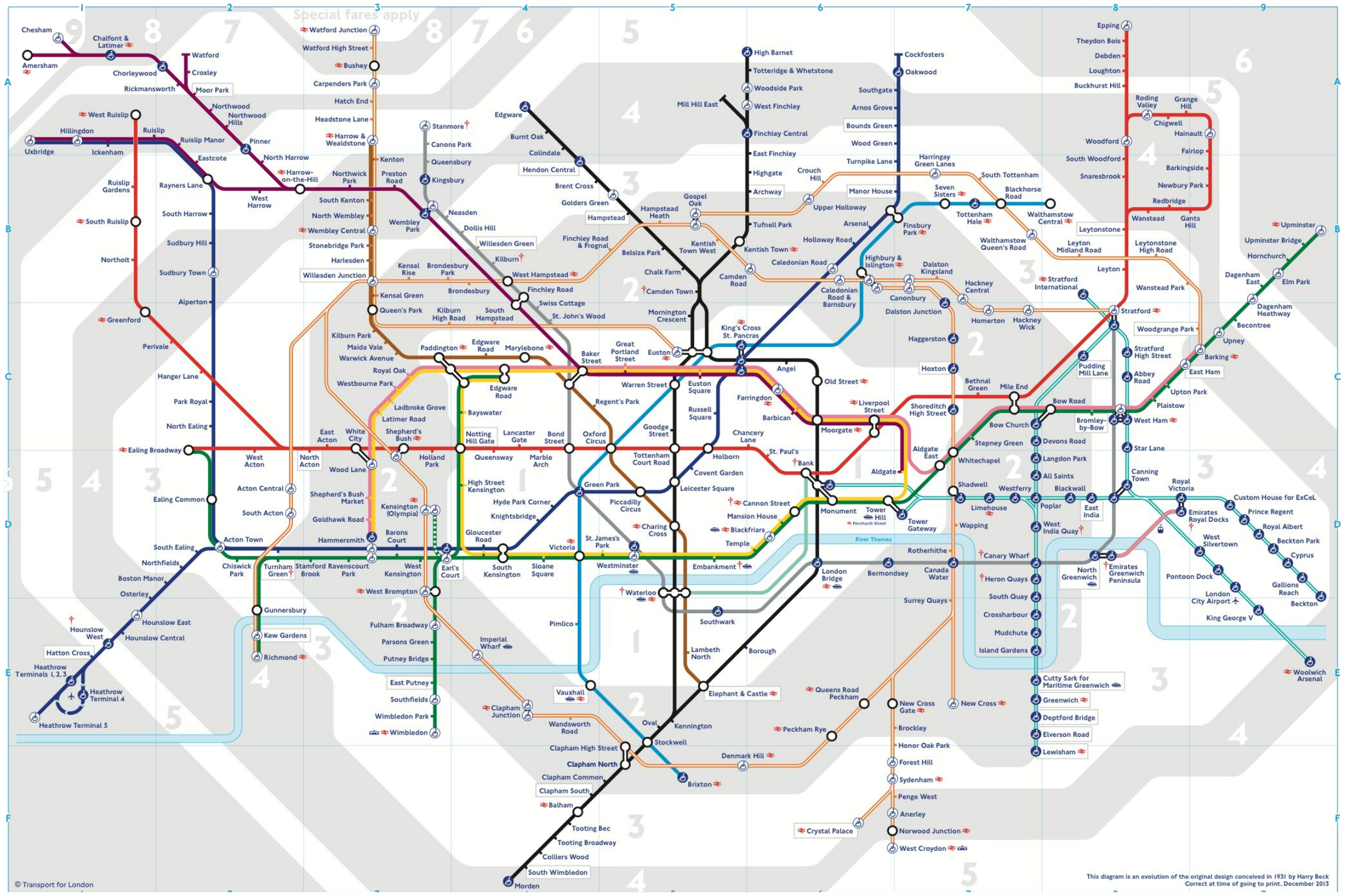
Visualisierung des Internets



Visualisierung von Facebook-Freunden

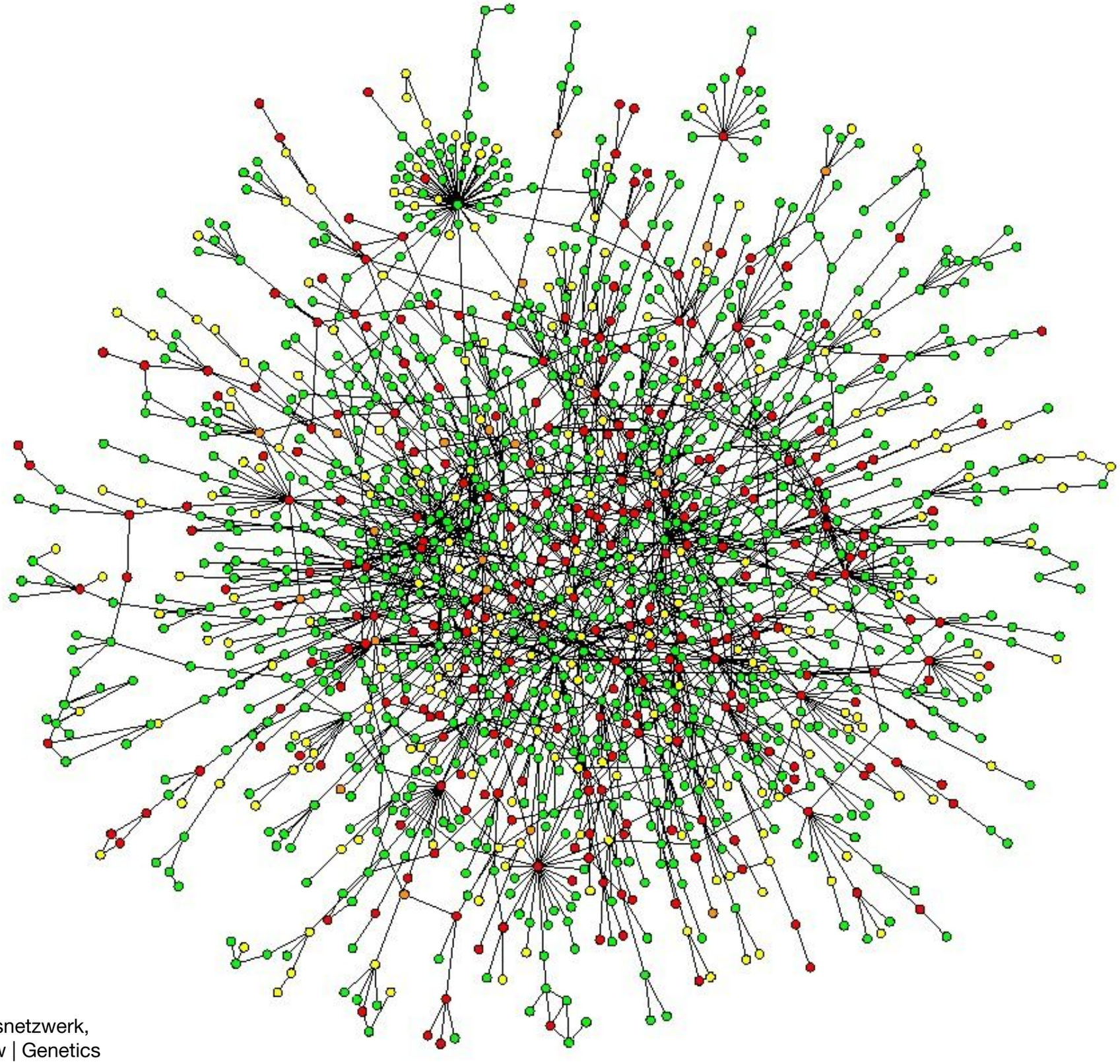
“Visualizing Facebook Friends” by Paul Butler
https://paulbutler.org/2010/visualizing-facebook-friends/

London Metro



London metro, London Transport

Netzwerke von Proteininteraktionen



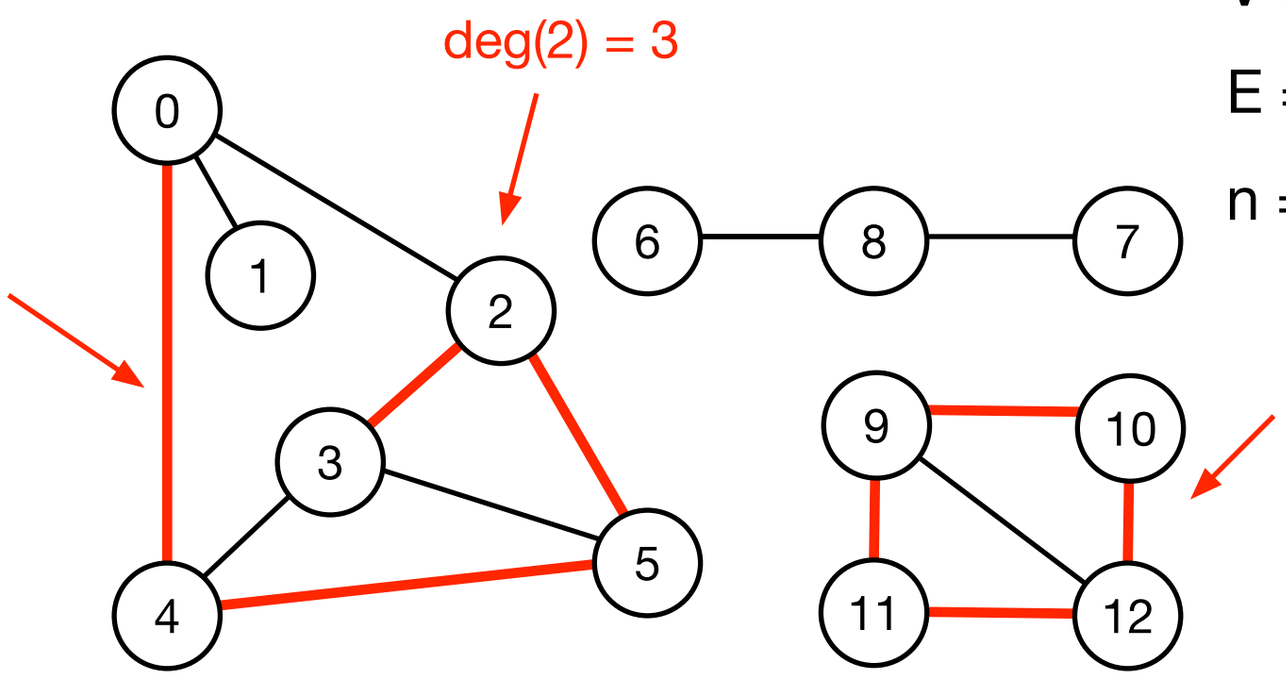
Protein-Protein Interaktionsnetzwerk,
Jeong et al., Nature Review | Genetics

Anwendungen von Graphen

Graph	Knoten	Kanten
Kommunikation	Computer	Kabel
Straßennetz	Kreuzungen	Straßen
Flugverkehr	Flughäfen	Flugverbindungen
(Brett)spiele	Stellung	gültige Züge
Neuronale Netze	Neuronen	Synapsen
Finanznetzwerke	Aktien	Transaktionen
Schaltkreise	logische Gatter	Verbindungen
Nahrungskette	Spezies	Raubtier-Beute
Moleküle	Atome	Bindungen

Terminologie

- Ungerichteter Graph. G = (V, E)
  - V = Menge von Knoten
  - E = Menge von Kanten (jede Kante ist ein **ungeordnetes** Paar (u,v) von Knoten)
  - n = |V|, m = |E|
- Weg. Sequenz von Knoten, die durch Kantenzug verbunden sind.
- Kreis. Weg der am selben Knoten beginnt und endet.
- Grad. deg(v) = die Anzahl der Nachbarn von v, oder Kanten inzident zu v.
- Verbunden. Zwei Knoten sind verbunden, wenn es einen Weg zwischen ihnen gibt.



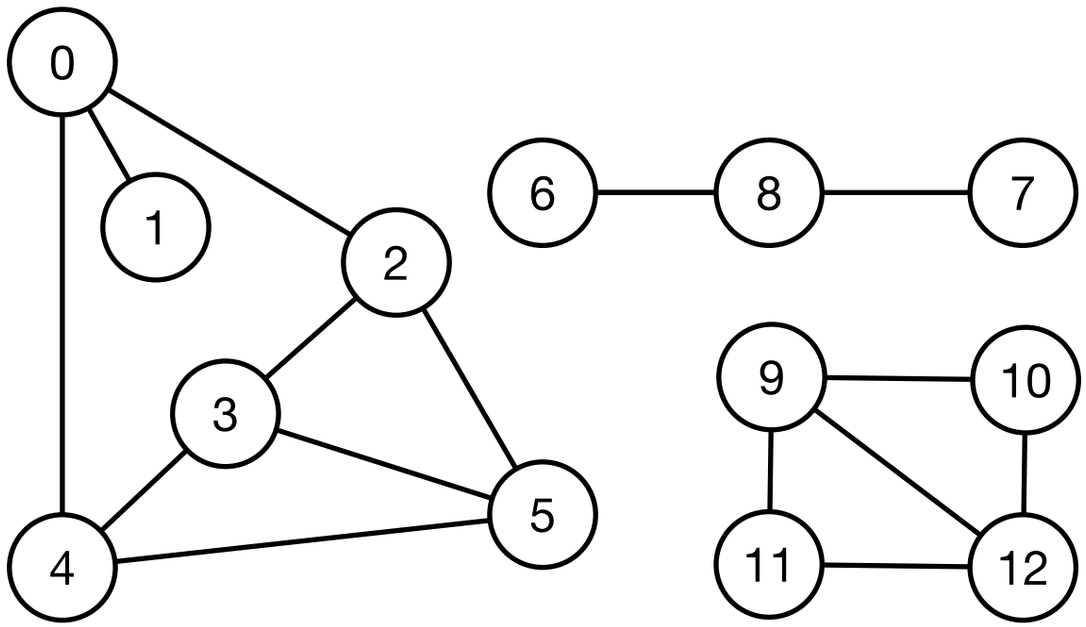
V = {0,1,2,…,12}

E = {(0,1), (0,2) (0,4),(2,3),…, (11,12)}

n = 13, m = 15

Ungerichtete Graphen

- Lemma. $\sum_{v \in V} \deg(v) = 2m$.
- Beweis. Wie oft wird jede Kante in der Summe gezählt?



Algorithmische Probleme auf Graphen

- Weg. Gibt es einen Weg, der s und t verbindet?
- Kürzester Weg. Was ist der kürzeste Weg von s nach t?
- Längster. Was ist der längste Weg von s nach t?

- Kreis. Gibt es einen Kreis im Graphen?
- Eulerkreis. Gibt es einen Kreis, der jede Kante genau einmal benutzt?
- Hamiltonkreis. Gibt es einen Kreis, der jeden Knoten genau einmal benutzt?

- Verbundenheit. Sind alle Paar von Knoten miteinander verbunden?
- Minimaler Spannbaum. Was ist der beste Weg, alle Knoten zu verbinden?
- 2-Zusammenhang. Gibt es einen Knoten, durch dessen Löschung der Graph nicht mehr zusammenhängend ist?

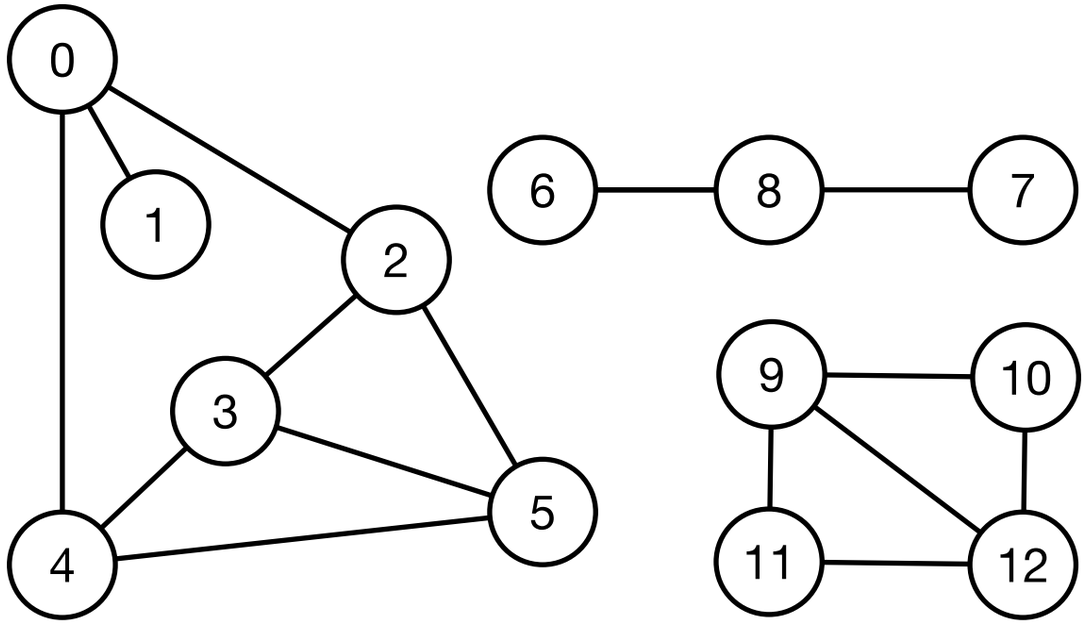
- Planarität. Lässt sich Graph in der Ebene zeichnen, ohne dass Kanten sich kreuzen?
- Graphisomorphie. Repräsentieren diese Mengen von Knoten und Kanten denselben Graph?

Einführung zu Graphen

- Ungerichtete Graphen
- Darstellung
- Tiefensuche
    - Zusammenhangskomponenten
- Breitensuche
    - Bipartite Graphen

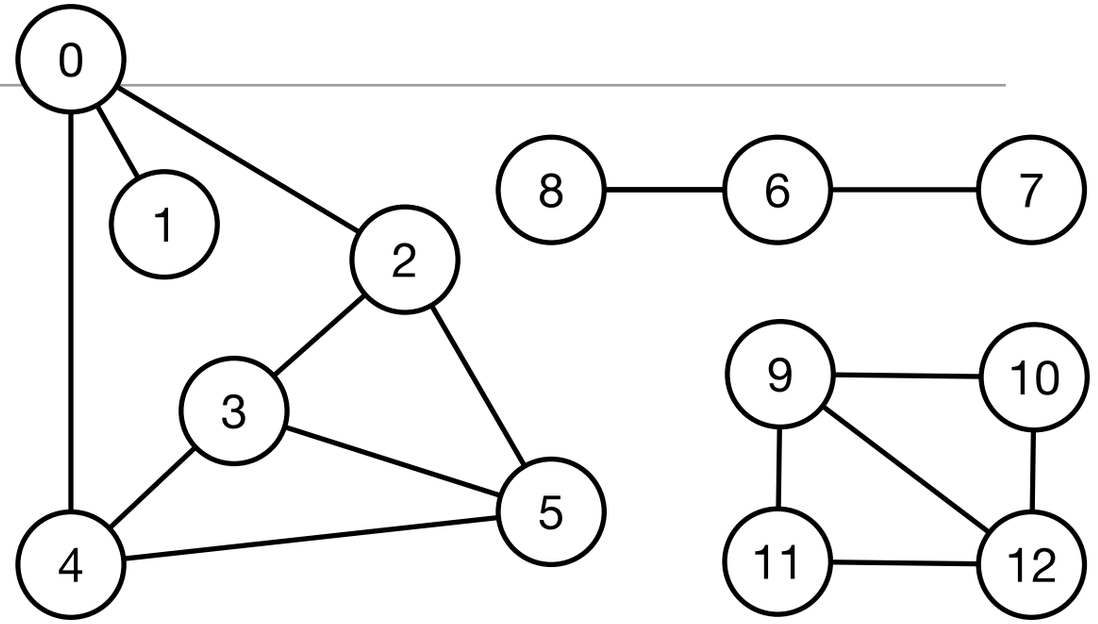
Darstellung

- Graph G mit n Knoten und m Kanten.
- Darstellung. Wir brauchen die folgenden Operationen auf Graphen.
  - ADJACENT(v, u): stelle fest, ob u und v benachbart sind.
  - NEIGHBORS(v): liefere alle Nachbarn von v zurück.
  - INSERT(v, u): füge die Kante (v, u) in G ein (sofern sie noch nicht existiert).



Adjazenzmatrix

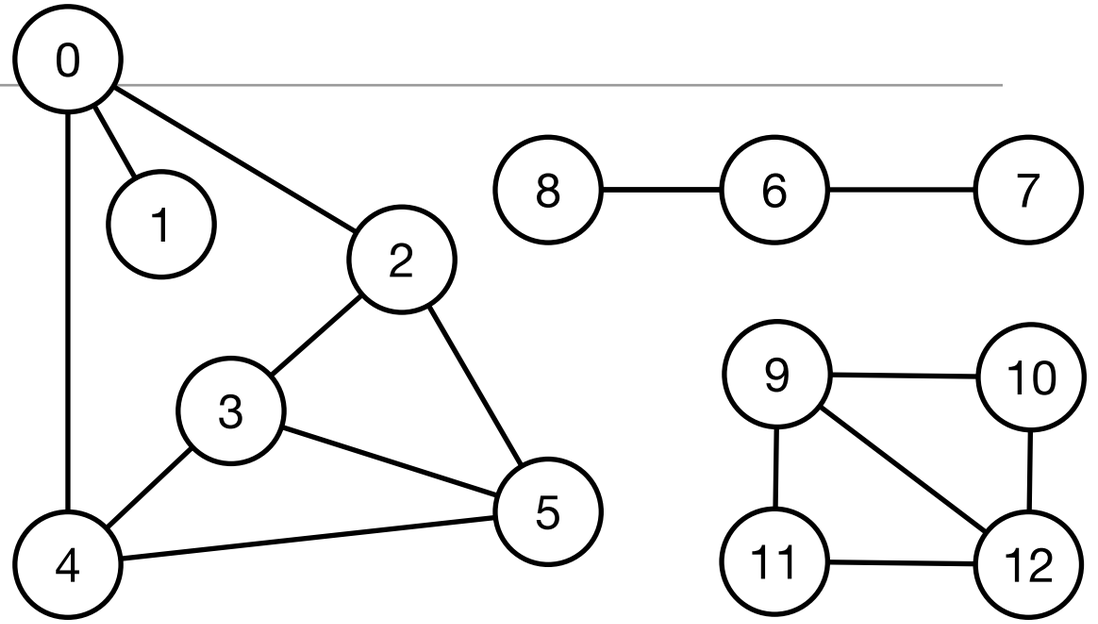
- Graph G mit n Knoten und m Kanten.
- Adjazenzmatrix.
  - 2D (n x n) Feld A.
  - A[i,j] = 1 wenn i und j benachbart, 0 sonst
- Komplexität?
- Platz. $O(n^2)$
- Zeit.
  - ADJACENT und INSERT in Zeit O(1).
  - NEIGHBOURS in Zeit O(n).

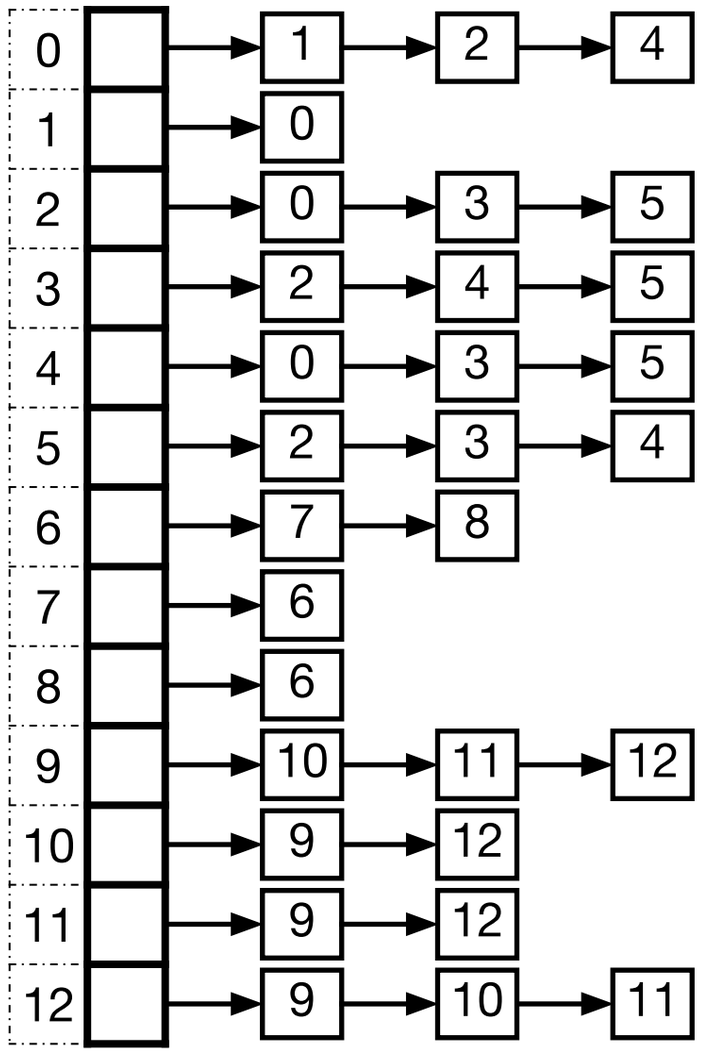


	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	1	0	0	0	0	0	0	0
3	0	0	1	0	1	1	0	0	0	0	0	0	0
4	1	0	0	1	0	1	0	0	0	0	0	0	0
5	0	0	1	1	1	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	1	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1
10	0	0	0	0	0	0	0	0	0	1	0	0	1
11	0	0	0	0	0	0	0	0	0	1	0	0	1
12	0	0	0	0	0	0	0	0	0	1	1	1	0

Adjazenzliste

- Graph G mit n Knoten und m Kanten.
- Adjazenzliste
  - Feld A[0..n-1].
  - A[i] ist eine verkettete Liste aller Nachbarn von i.
- Komplexität?
- Platz. $O(n + \sum_{u \in V} \deg(u)) = O(n + m)$
- Zeit.
  - ADJACENT, NEIGHBOURS, INSERT in Zeit O(deg(v)).





Darstellung

Datenstruktur	ADJACENT	NEIGHBOURS	INSERT	space
Adjazenzmatrix	O(1)	O(n)	O(1)	$O(n^2)$
Adjazenzliste	O(deg(v))	O(deg(v))	O(deg(v))	O(n+m)

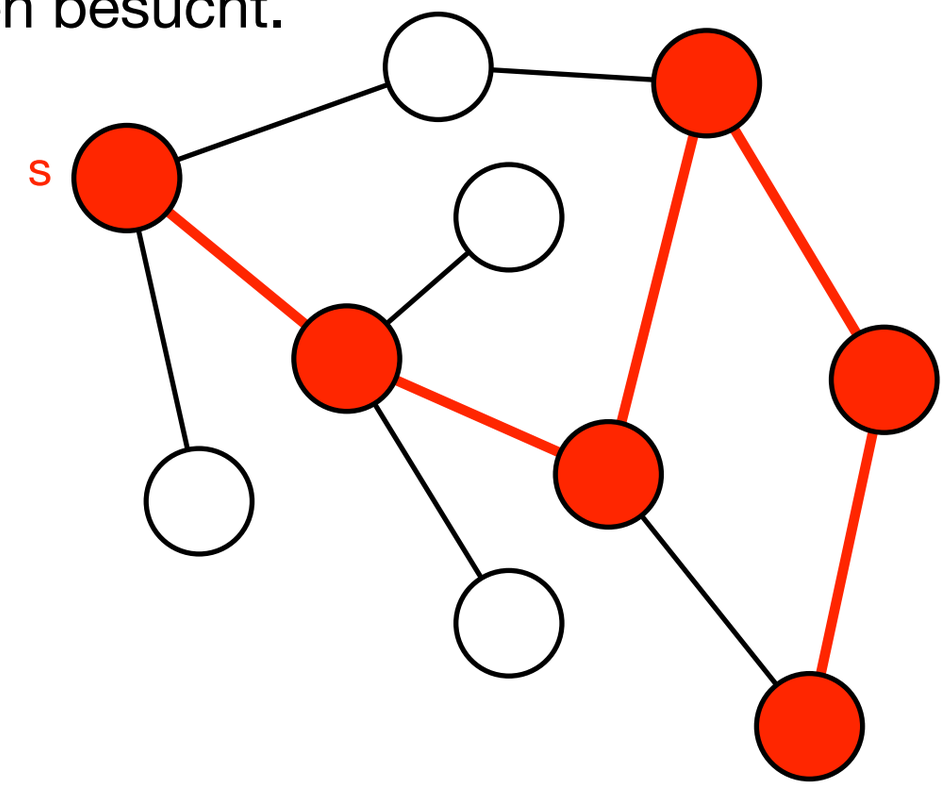
- In der echten Welt sind Graphen oft dünn.

Einführung zu Graphen

- Ungerichtete Graphen
- Darstellung
- Tiefensuche
  - Zusammenhangskomponenten
- Breitensuche
  - Bipartite Graphen

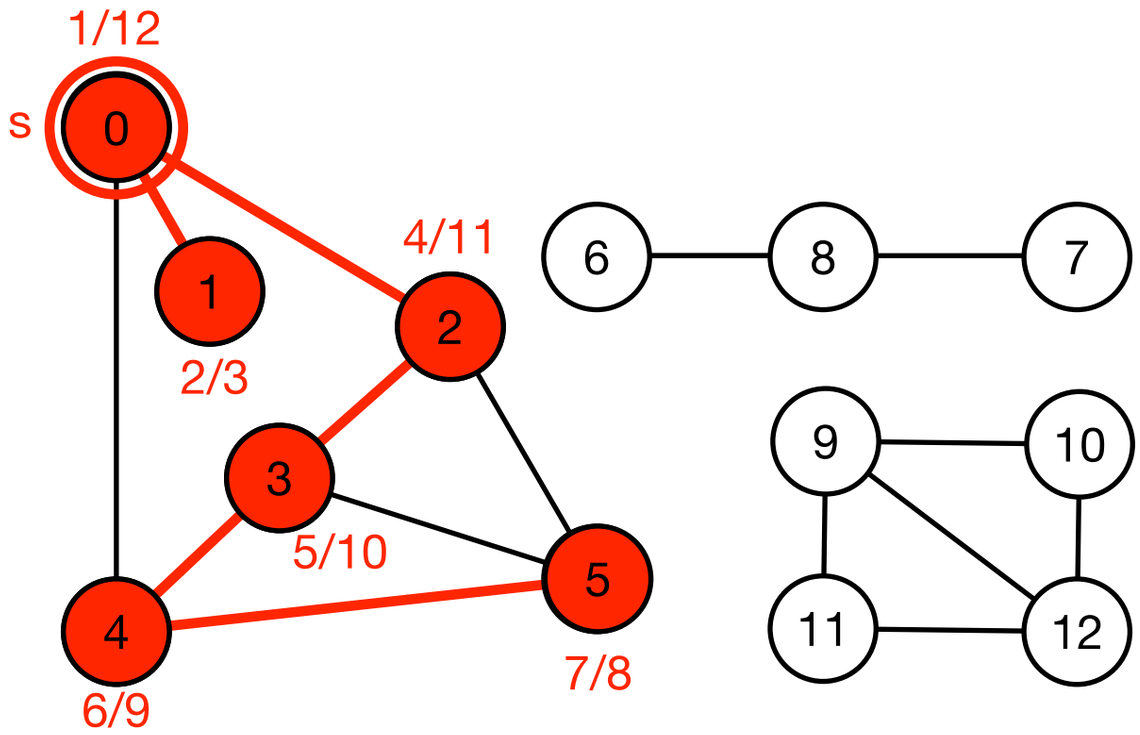
Tiefensuche

- Algorithmus, der systematisch alle Knoten und Kanten besucht.
- Tiefensuche von Knoten s.
  - Entferne alle Markierungen und besuche s.
  - Um Knoten v zu besuchen:
    - Markiere v.
    - Besuche alle unmarkierten Nachbarn von v rekursiv.



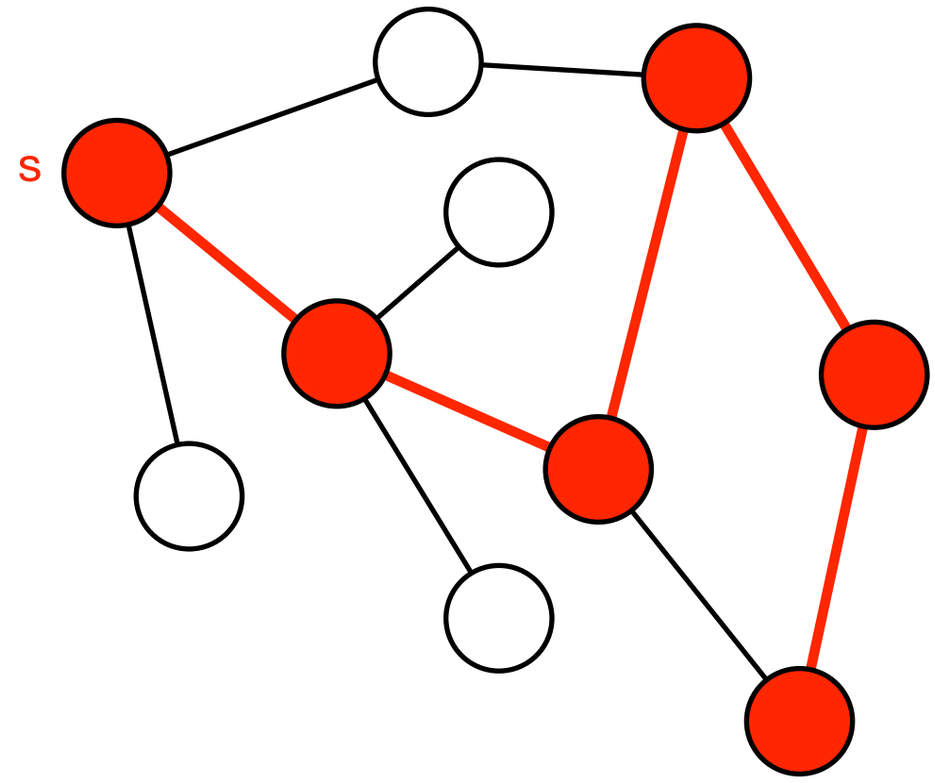
- Intuition.
  - Suche von s ausgehend in eine Richtung, bis wir eine Sackgasse erreichen.
  - Geh zurück zur letzten Position, die unbesuchte Kanten hat.
  - Wiederhole.

- Entdeckungszeitpunkt. Erster Zeitpunkt, zu dem ein Knoten besucht wird.
- Endzeitpunkt. Letzter Zeitpunkt, zu dem ein Knoten besucht wird.


1/12
s
0
1
2
4/11
2/3
3
5/10
4
5
6/9
7/8
6
8
7
9
10
11
12

Tiefensuche (*Depth-first Search*)

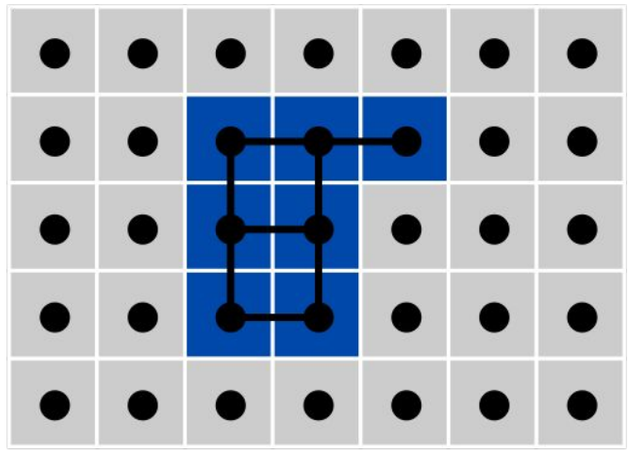
```
DFS(s)
   zeit = 0
   DFS-Visit(s)

DFS-Visit(u)
   u.d = ++zeit
   markiere u
   für jeden unmarkierten Nachbarn v:
      v.π = u
      DFS-Visit(v)
   u.f = ++zeit
```



- Zeit. (wenn G als Adjazenzliste gegeben ist)
  - Rekursion? Einmal pro Knoten.
  - O(deg(v)) Zeit wird pro Knoten v aufgewendet.
  - $\Rightarrow$ Gesamtzeit $O(n + \sum_{v \in V} \deg(v)) = O(n + m)$.
  - Besucht nur Knoten, die von s erreichbar sind.

Flood Fill

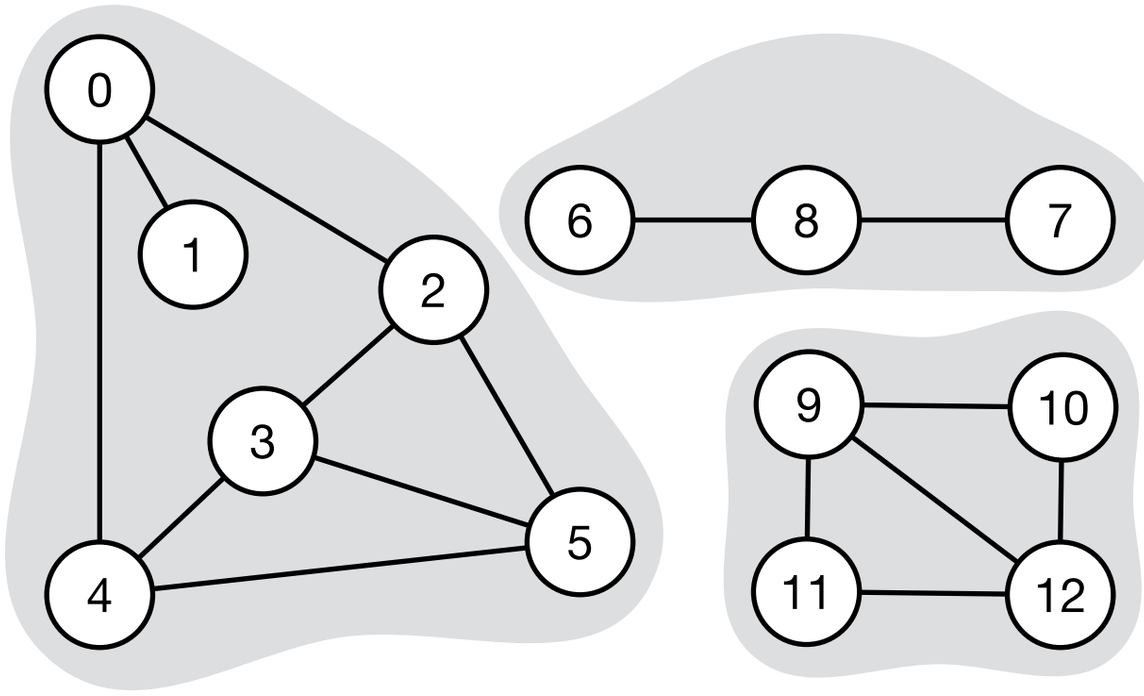
- Flood fill. Verändere die Farbe einer zusammenhängenden Fläche.



- Algorithmus.
  - Baue einen Gittergraphen und lass DFS laufen.
  - Knoten: Pixel.
  - Kanten: Zwischen benachbarten Pixeln derselben Farbe.
  - Fläche: Zusammenhangskomponente

Zusammenhangskomponenten

- Definition. Eine Zusammenhangskomponente ist eine maximale Teilmenge von zusammenhängenden Knoten.



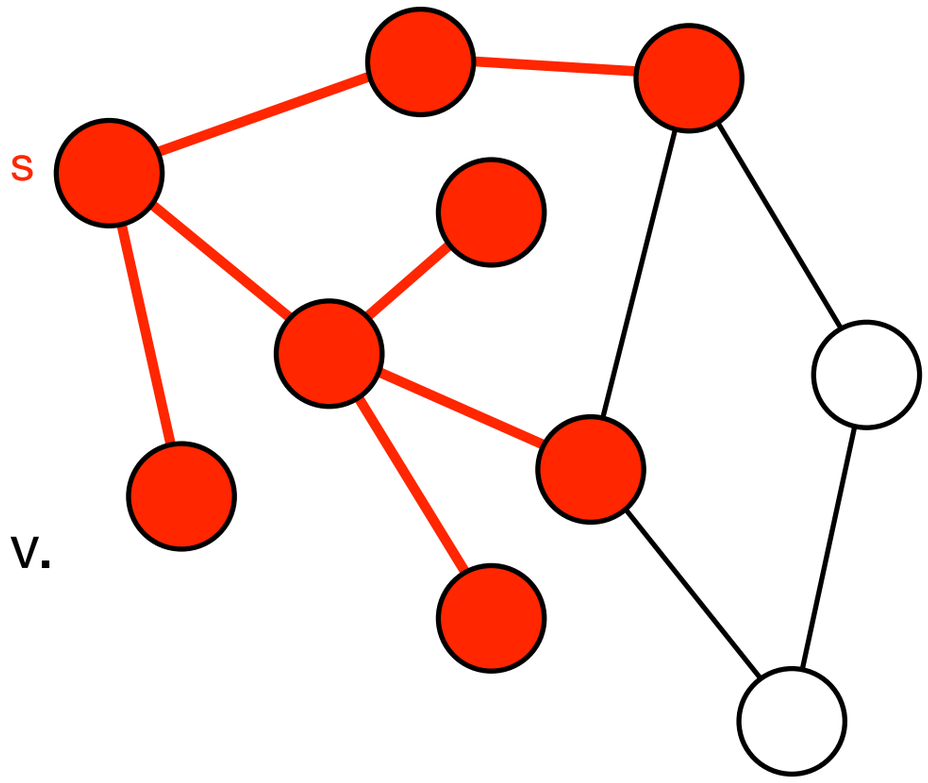
- Wie findet man alle Zusammenhangskomponenten?
- Algorithmus.
  - Entferne alle Markierungen
  - Solange es einen unmarkierten Knoten v gibt:
    - Lass DFS von v aus laufen.
- Zeit. O(n + m).

Einführung zu Graphen

- Ungerichtete Graphen
- Darstellung
- Tiefensuche
  - Zusammenhangskomponenten
- Breitensuche
  - Bipartite Graphen

Breitensuche (*breadth-first search*)

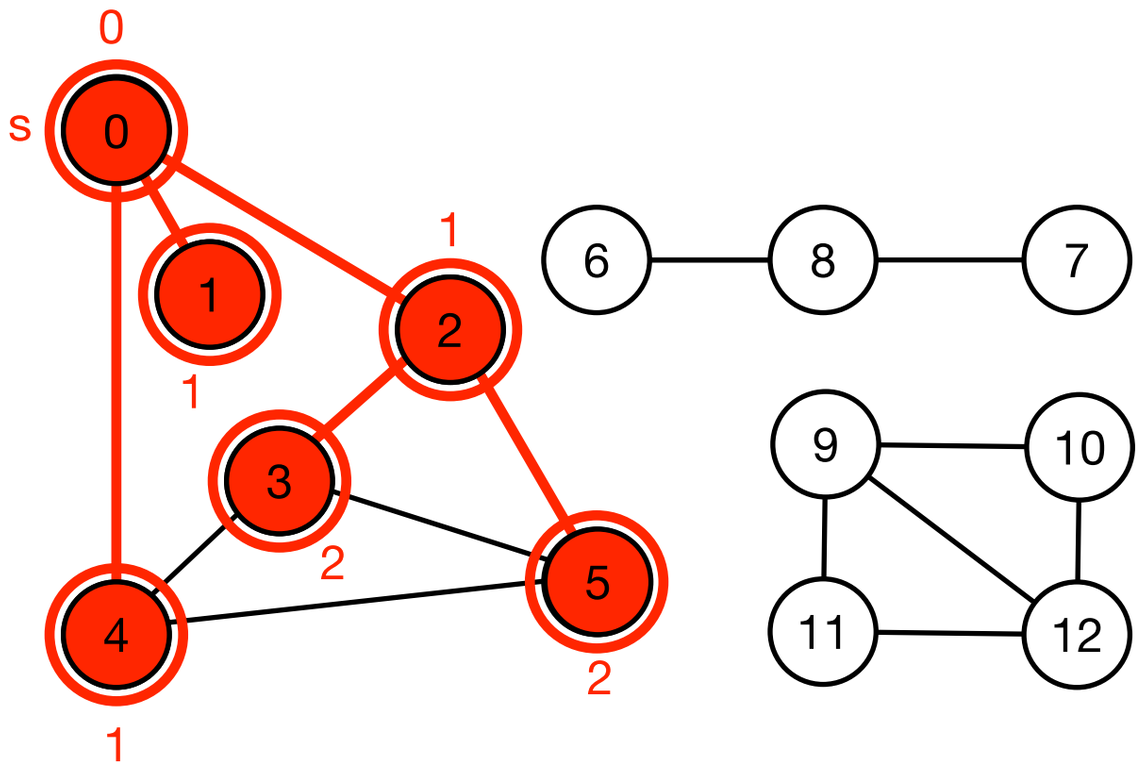
- Breitensuche von s.
  - Entferne alle Markierungen und initialisiere eine Schlange Q.
  - Markiere s und Q.ENQUEUE(s).
  - Solange Q nicht leer ist:
    - v = Q.DEQUEUE().
    - Für jeden unmarkierten Nachbarn u von v.
      - Markiere u.
      - Q.ENQUEUE(u).



- Intuition.
  - Suche startet von s aus in alle Richungen - im immer größeren Abstand zu s.

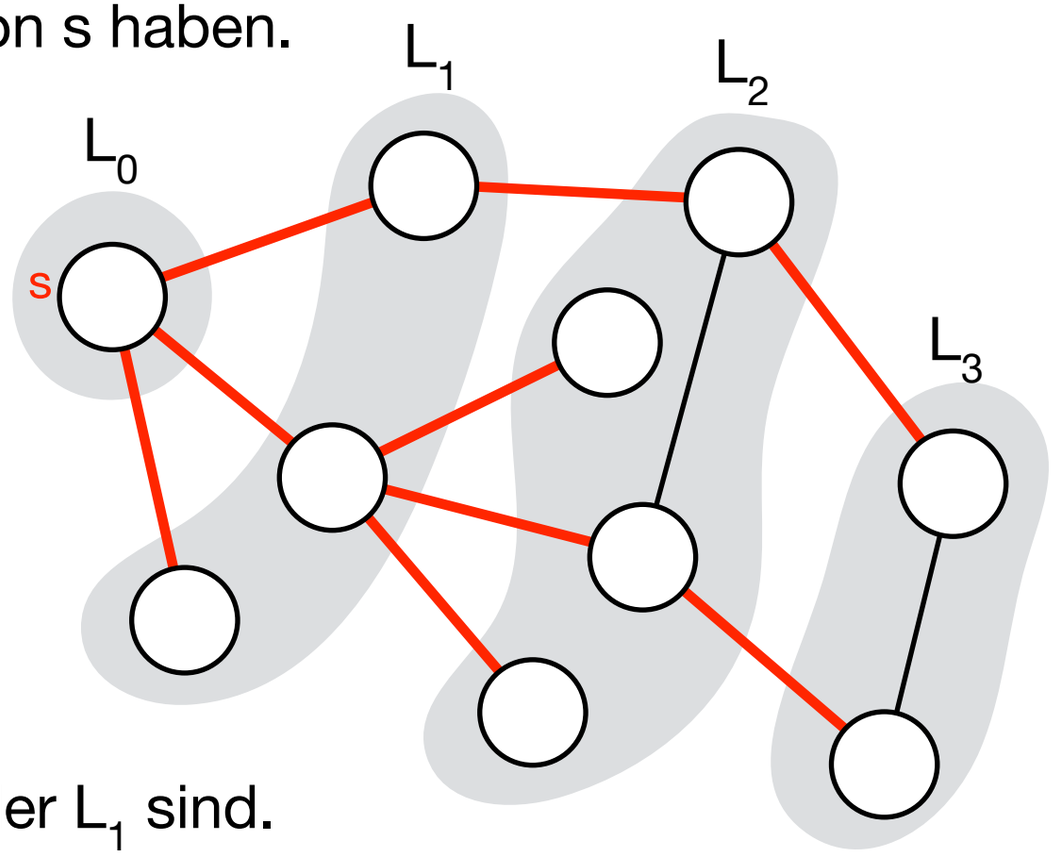
- Kürzeste Distanz zu s.
  - Abstand von s im BFS Baum = kürzeste Distanz zu s im Eingabegraphen.

Q 0 1 2 4 3 5



Kürzeste Pfade

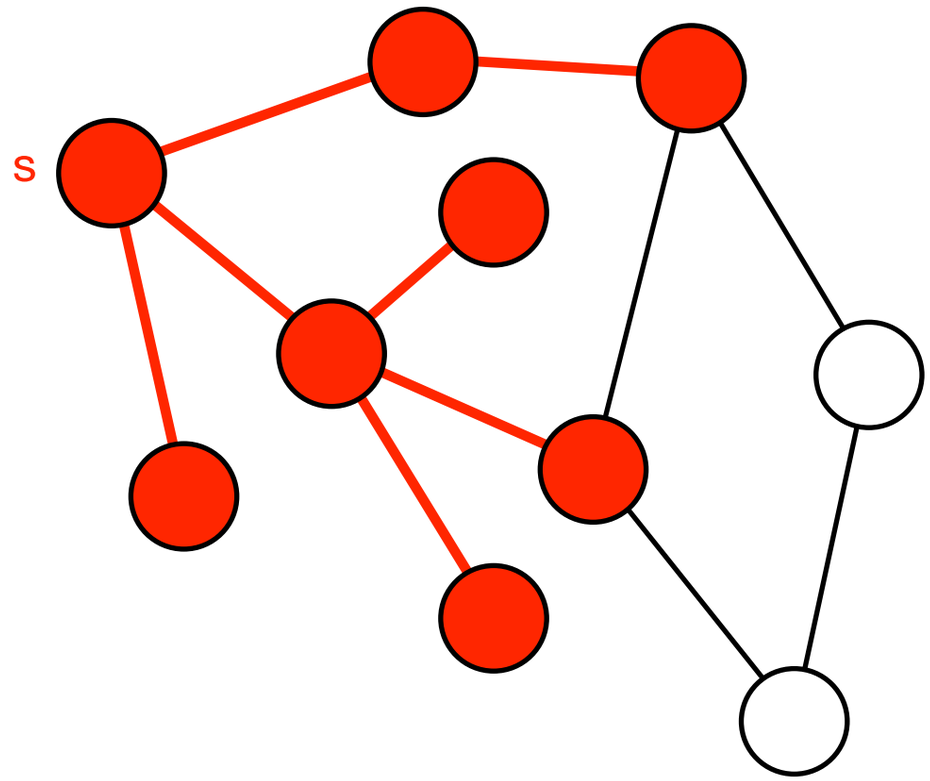
- Lemma. BFS findet die Länge des kürzesten Pfades von s zu allen anderen Knoten.
- Intuition.
  - BFS ordnet jedem Knoten eine Ebene zu.
    Ebene i enthält alle Knoten, die Distanz i von s haben.



  - Was enthält jede Ebene?
  - L_0 : {s}
  - L_1 : alle Nachbarn von L_0.
  - L_2 : alle Nachbarn von L_1, die nicht in L_0 oder L_1 sind.
  - L_3 : alle Nachbarn von L_2 , die nicht in L_0, L_1 oder L_2 sind.
  - ...
  - L_i : alle Nachbarn von L_{i-1}, die nicht in L_0, ..., L_{i-1} sind.
    = alle Knoten von Distanz i zu s.

Breitensuche

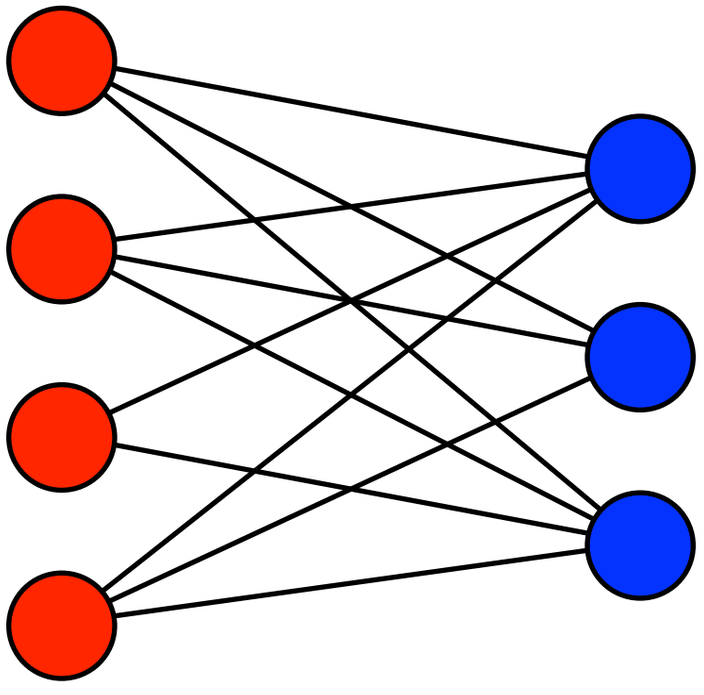
```
BFS(s)
   markiere s
   s.d = 0
   Q.ENQUEUE(s)
   repeat until Q.ISEMPTY()
      v = Q.DEQUEUE()
      für jeden unmarkierten Nachbarn u
         markiere u
         u.d = v.d + 1
         u.π = v
         Q.ENQUEUE(u)
```



- Zeit. (wenn G als Adjazenzliste gegeben ist)
  - Jeder Knoten wird höchstens einmal besucht.
  - O(deg(v)) Zeitaufwand pro Knoten v.
  - $\Longrightarrow$ Gesamtzeit $O(n + \sum_{v \in V} \deg(v)) = O(n + m)$.
  - Nur Knoten, die von s auch erreichbar sind werden besucht.

Bipartite Graphen

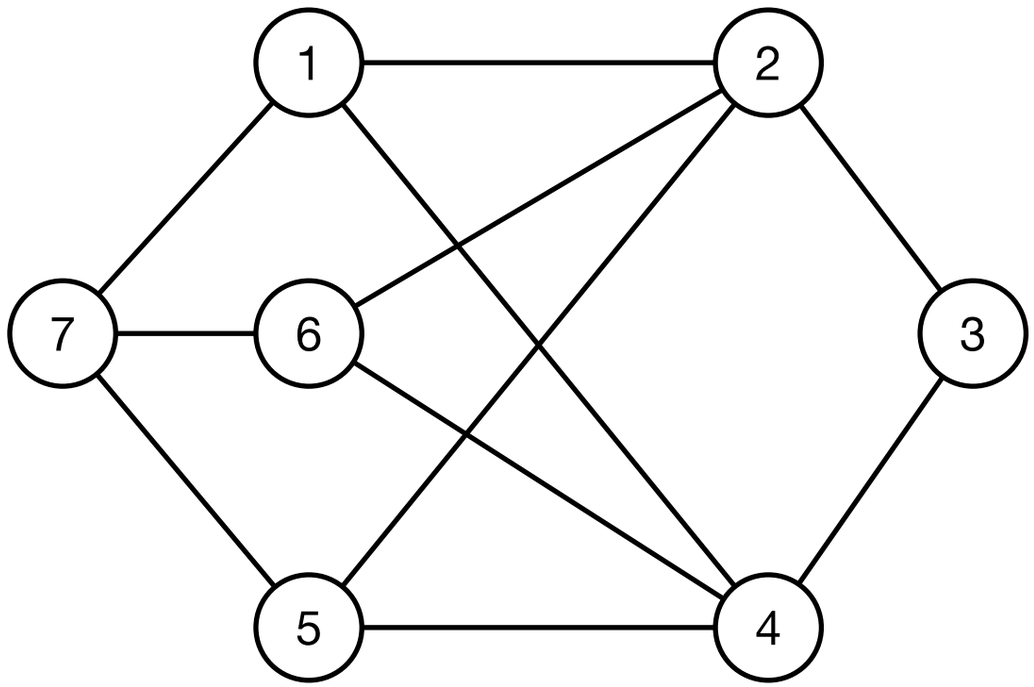
- Definition. Ein Graph ist bipartit genau dann, wenn alle Knoten rot oder blau gefärbt werden können, sodass **alle** Kanten einen roten Endpunkt und einen blauen Endpunkt haben.
- Äquivalente Definition. Ein Graph ist bipartit genau dann, wenn seine Knoten in zwei Mengen V_1 und V_2 partitioniert werden können, sodass alle Kanten zwischen V_1 und V_2 verlaufen.

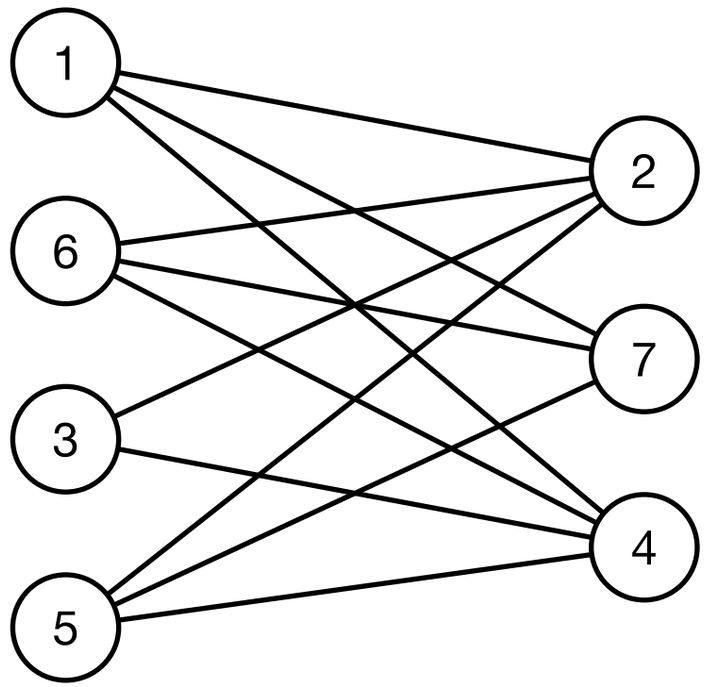


- Anwendungen.
  - Scheduling, matching, Referendar:innen auf Schulen verteilen, Student:innen auf Unis verteilen, Jobs auf Maschinen verteilen, …
  - Viele Graphprobleme sind *einfacher* auf bipartiten Graphen.

Bipartite Graphen

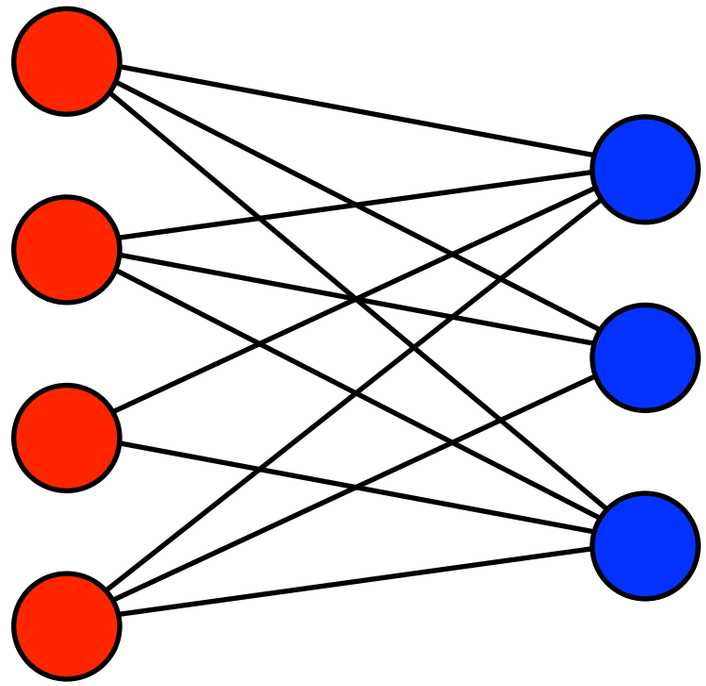
- Frage. Gegeben ein Graph G, finde heraus ob G bipartit ist.





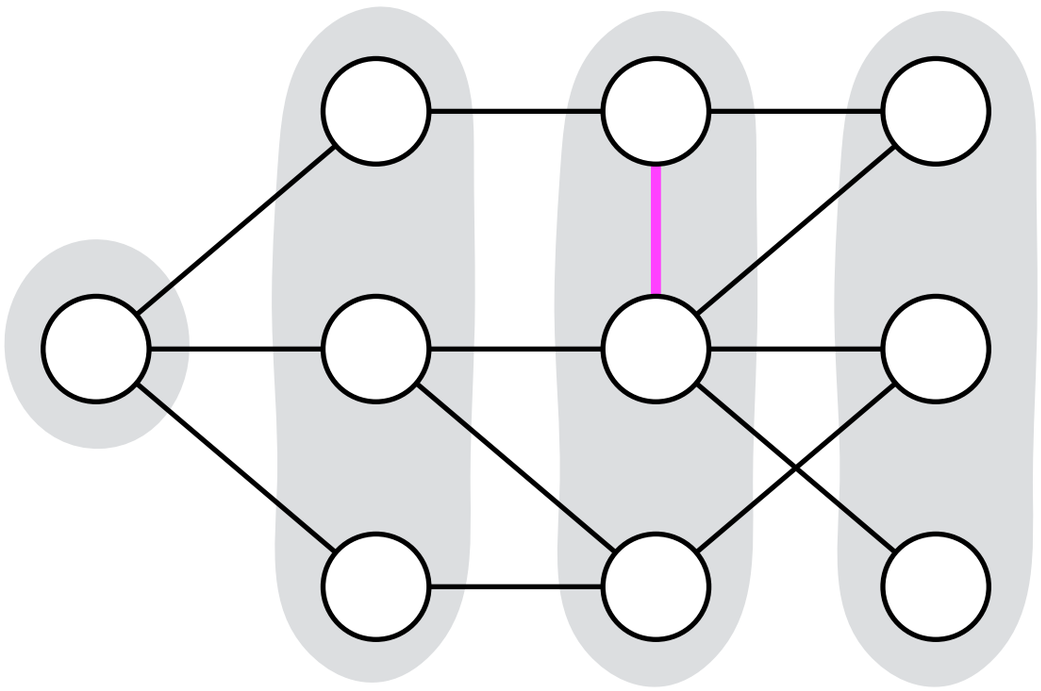
Bipartite Graphen

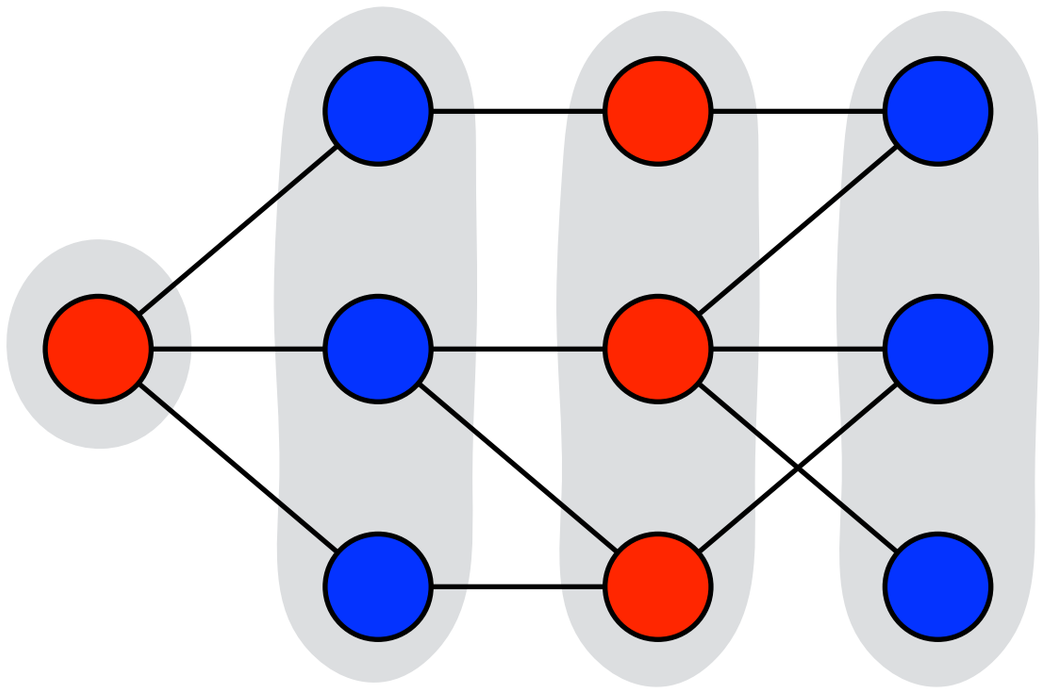
- Lemma. Ein Graph G ist bipartit genau dann, wenn alle Kreise von G gerade Länge haben.
- Beweis. $\Rightarrow$
  - Wenn G bipartit ist, starten und enden alle Kreise auf derselben Seite.



Bipartite Graphen

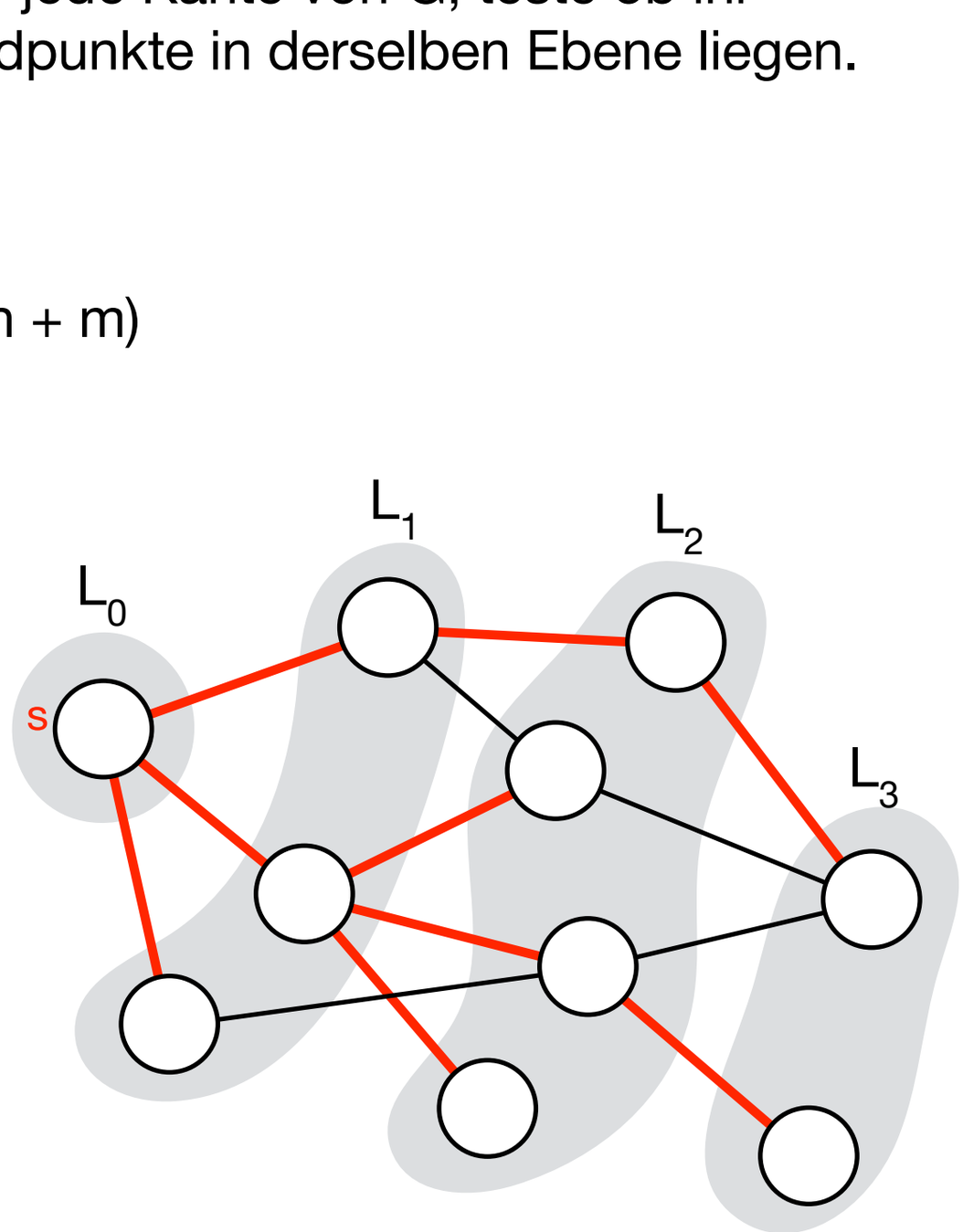
- Lemma. Ein Graph G ist bipartit genau dann, wenn alle Kreise von G gerade Länge haben.
- Beweis. $\Leftarrow$
  - Wähle einen Knoten v und betrachte die BFS Ebenen L_0, L_1, …, L_k.
  - Alle Kreise haben gerade Länge.
  - $\Rightarrow$ Es gibt keine Kante zwischen Knoten derselben Ebene.
  - $\Rightarrow$ Wir können die Ebenen abwechselnd rot oder blau färben.
  - $\Rightarrow$ G ist bipartit.

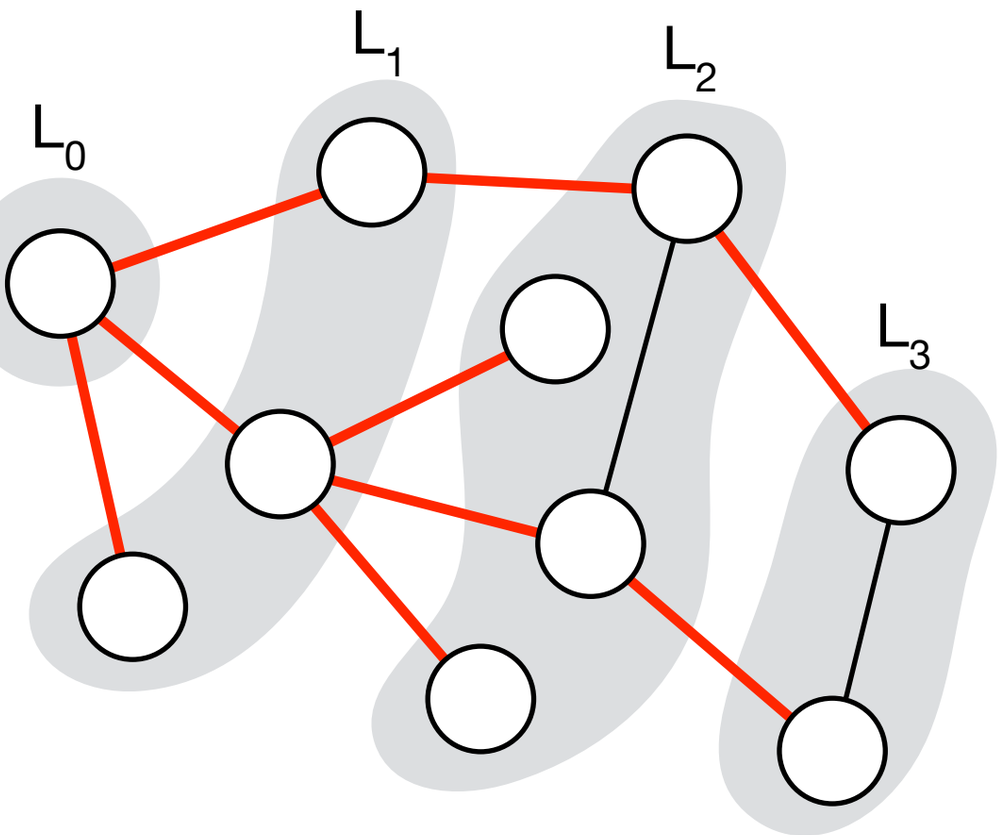




Bipartite Graphen

- Algorithmus.
  - Lass BFS auf G laufen.
  - Für jede Kante von G, teste ob ihr Endpunkte in derselben Ebene liegen.
- Zeit.
  - O(n + m)





Graphalgorithmen

Algorithmus	Zeit	Platz
Tiefensuche (DFS)	O(n + m)	O(n + m)
Breitensuche (BFS)	O(n + m)	O(n + m)
Zusammenhangskomponenten	O(n + m)	O(n + m)
Bipartit	O(n + m)	O(n + m)

- Alle Algorithmen hier erwarten die Graphen in Adjazenzlistendarstellung.

Einführung zu Graphen

- Ungerichtete Graphen
- Darstellung
- Tiefensuche
    - Zusammenhangskomponenten
- Breitensuche
    - Bipartite Graphen