Introduktion til grafer

- Uorienterede grafer
- Repræsentation
- Dybdeførst søgning
  - Sammenhængskomponenter
- Breddeførst søgning
  - Todelte grafer

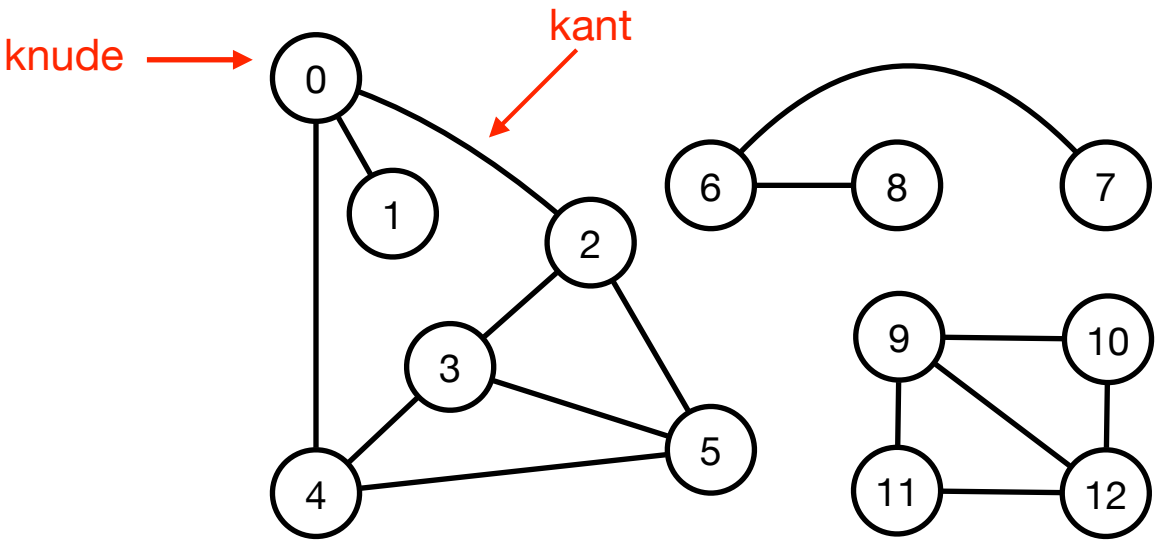
Philip Bille

Introduktion til grafer

- Uorienterede grafer
- Repræsentation
- Dybdeførst søgning
  - Sammenhængskomponenter
- Breddeførst søgning
  - Todelte grafer

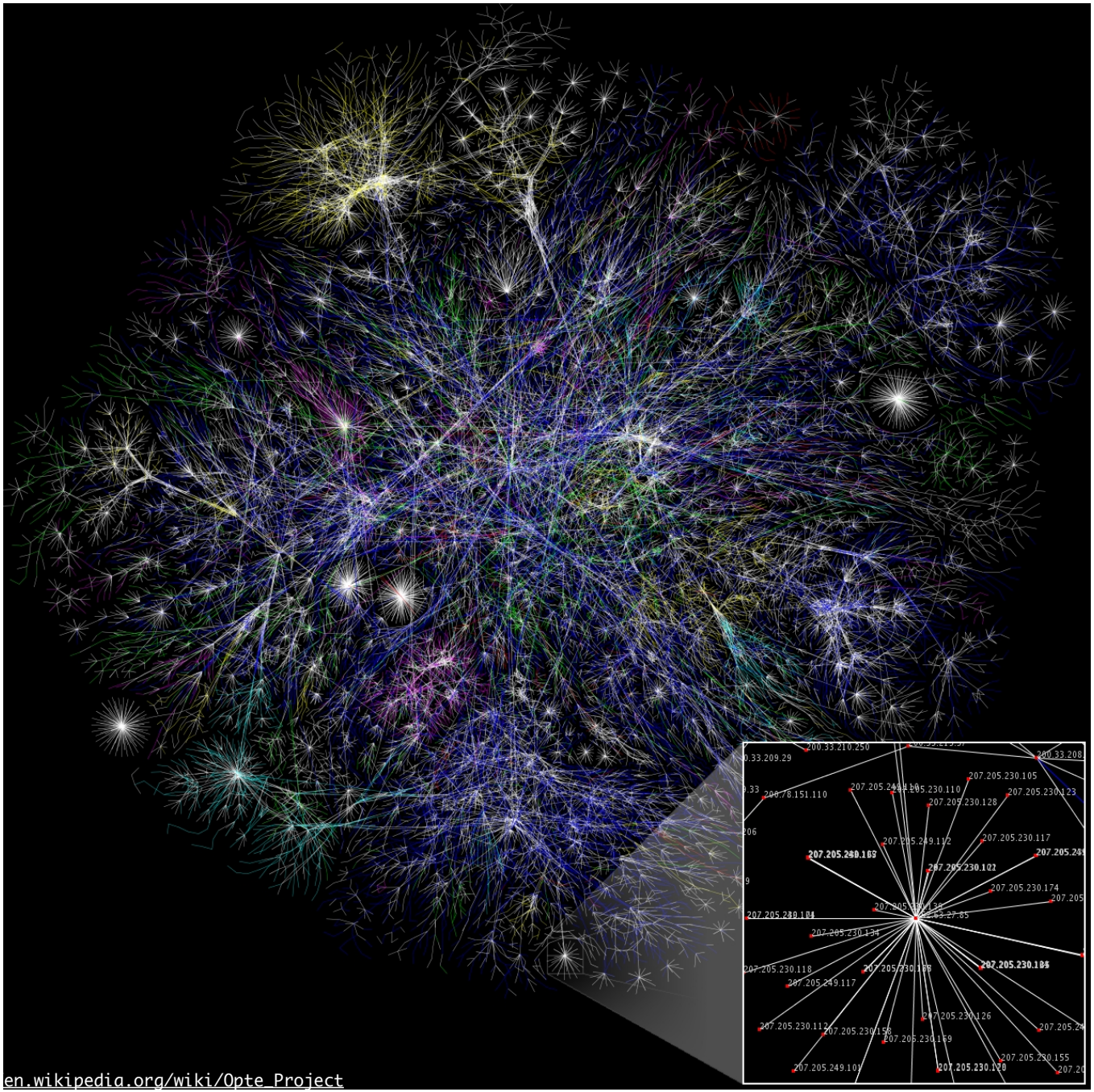
Uorienterede grafer

- Uorienteret graf *(undirected graph)*. Mængde af knuder *(vertices)* forbundet parvis med kanter *(edges)*.



- Hvorfor grafer?
  - Modellerer naturligt mange problemer i mange forskellige områder.
  - Tusindvis af praktiske anvendelser.
  - Hundredevis af kendte grafalgoritmer.

Visualisering af internettet

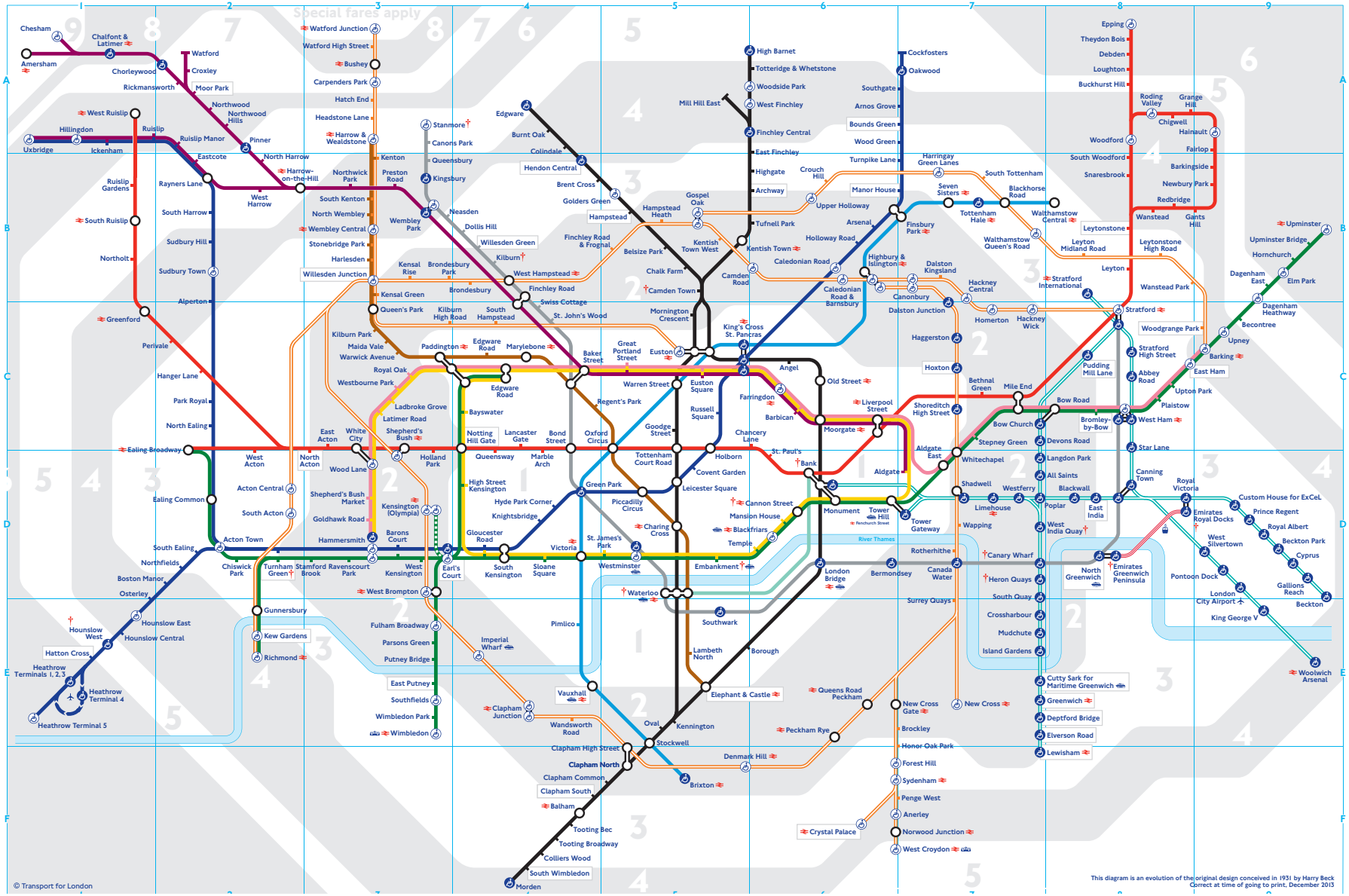


en.wikipedia.org/wiki/Opte_Project

Visualisering af Facebook venskaber

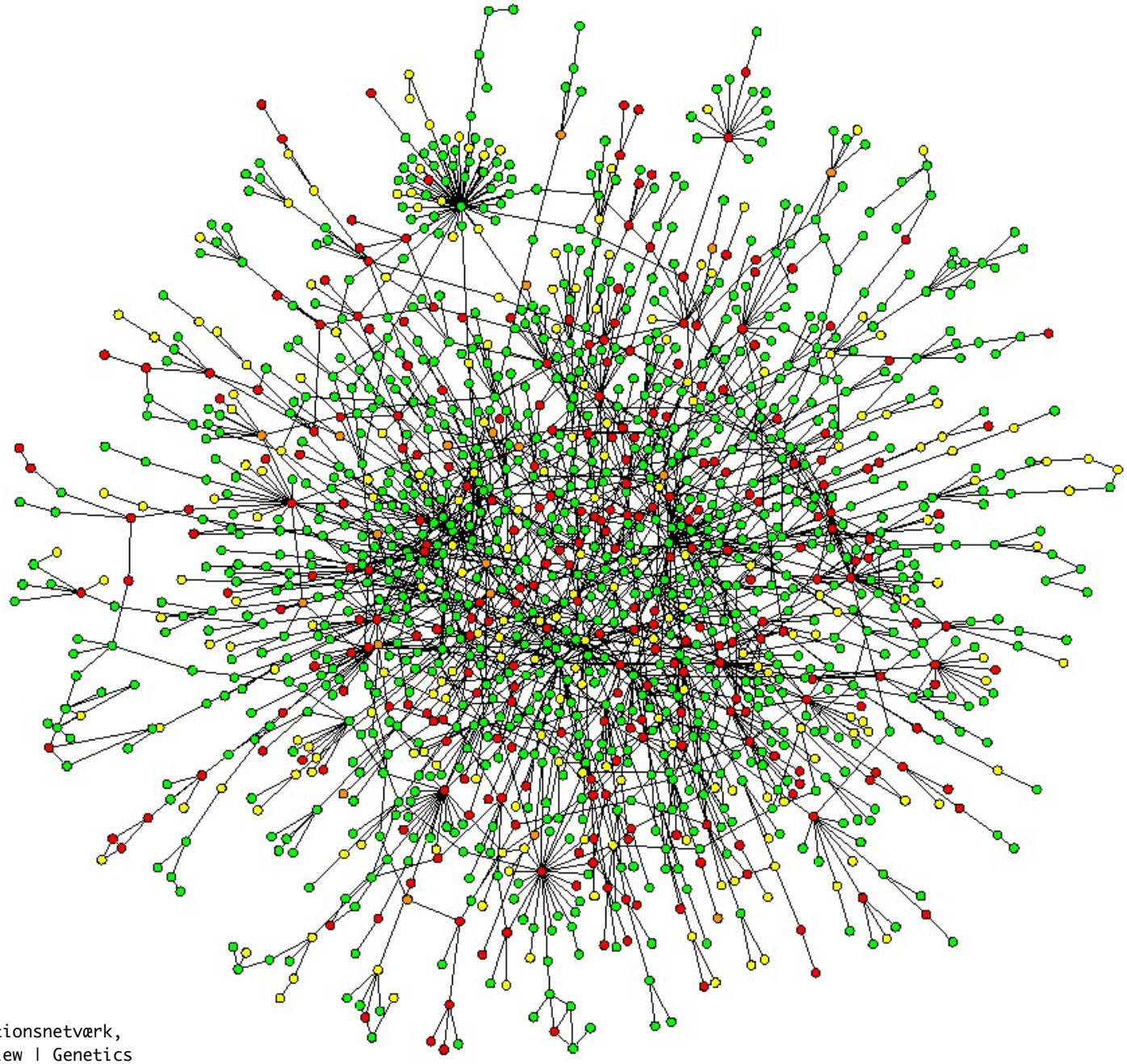
"Visualizing friendships", Paul Butler

London Metro



London metro, London Transport

Protein interaktionsnetværk



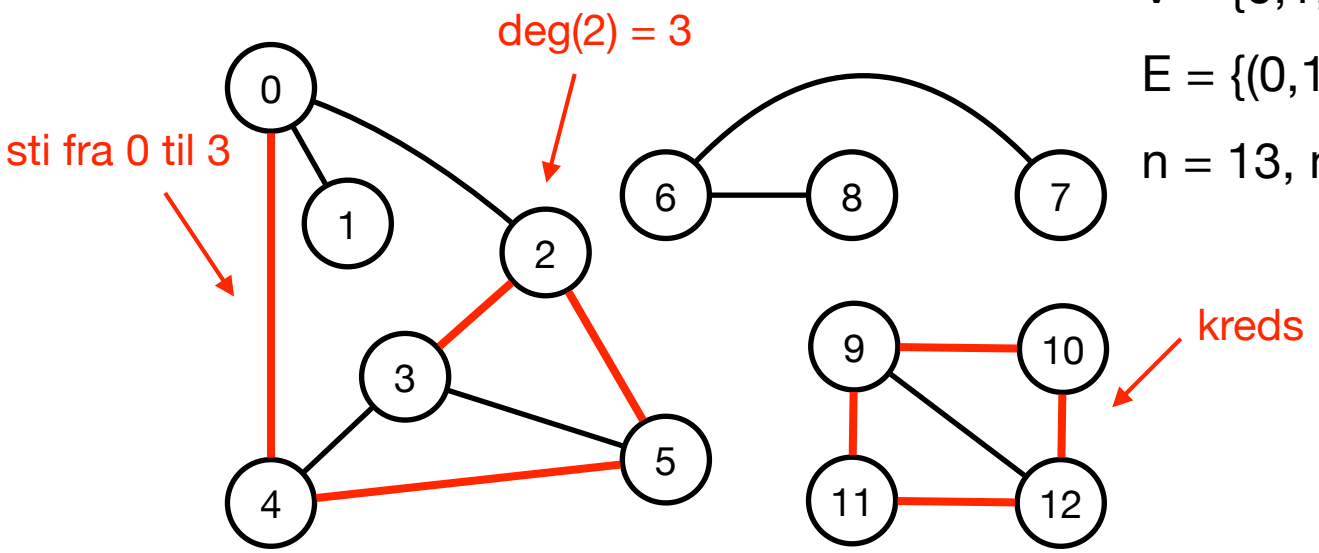
Protein-protein interaktionsnetværk,
Jeong et al, Nature Review | Genetics

Grafanvendelser

graf	knuder	kanter
kommunikation	computer	kabel
transport	vejkryds	vej
transport	lufthavn	fly
spil	position	lovligt træk
neuralt netværk	neuron	synapse
finansnetværk	valuta, aktier	transaktioner
kredsløb	logiske porte	forbindelse
fødekæde	dyrearter	rovdyr-byttedyr
molekyle	atom	bindinger

Terminologi

- Uorienteret graf. G = (V, E)
  - V = mængde af knuder
  - E = mængde af kanter (par af knuder)
  - n = |V|, m = |E|
- Sti/vej *(path)*. Sekvens af knuder forbundet af kanter.
- Kreds *(cycle)*. Sti hvis start- og slutknude er ens.
- Grad *(degree)*. deg(v) = antal naboer af v = antal kanter incidente med v.
- Sammenhæng *(connectivity)*. To knuder er forbundne hvis der er en vej i mellem dem



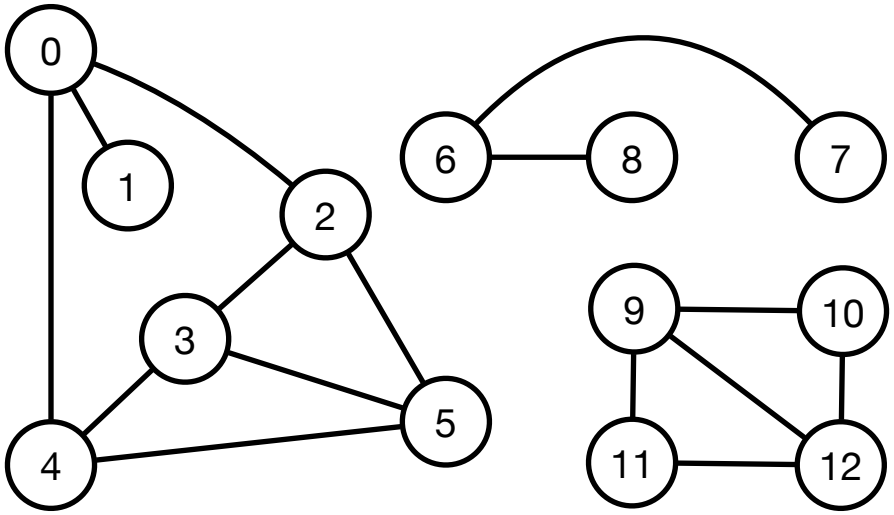
V = {0,1,2,…,12}

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

n = 13, m = 15

Uorienterede grafer

- Lemma. $\sum_{v \in V} \deg(v) = 2m$.
- Bevis. Hvor mange gange tælles kant med i sum?



Algoritmiske problemer på grafer

- Vej. Findes der en sti mellem s og t?
- Korteste vej. Hvad er den korteste vej mellem s og t?
- Længste vej. Hvad er den længste vej mellem s og t?

- Kreds. Er der en kreds i grafen?
- Eulertur. Er der en kreds, der benytter hver kant netop en gang?
- Hamiltontur. Er der en kreds, der benytter hver knude netop en gang?

- Sammenhæng. Er det muligt at forbinde alle knuder med en sti?
- Mindste udspændende træ. Hvad er den bedste måde at forbinde alle knuder på?
- Tosammenhæng. Er der en knude hvis fjernelse gør grafen usammenhængende?

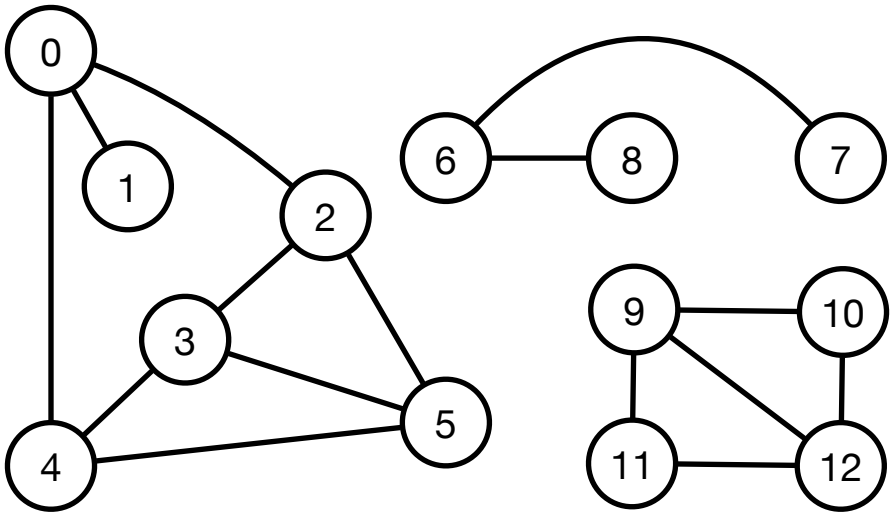
- Planaritet. Er det muligt at tegne grafen i planen uden kanter der krydser?
- Grafisomorfi. Repræsenterer to mængde af knuder og kanter samme graf?

Introduktion til grafer

- Uorienterede grafer
- Repræsentation
- Dybdeførst søgning
  - Sammenhængskomponenter
- Breddeførst søgning
  - Todelte grafer

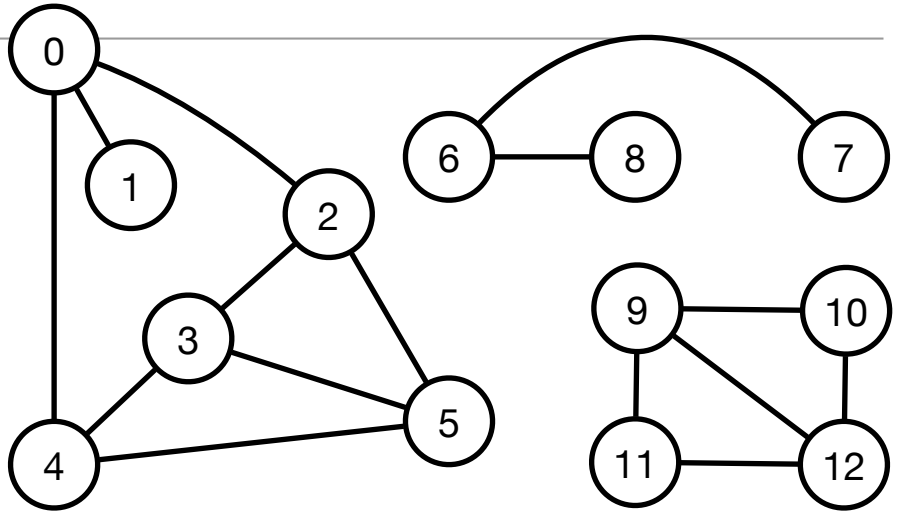
Repræsentation

- G graf med n knuder og m kanter.
- Repræsentation. Vi skal bruge følgende operationer på grafer.
  - ADJACENT(v, u): afgør om knude v og u er naboer.
  - NEIGHBORS(v): returner alle naboer af v.
  - INSERT(v, u): tilføj kant (v, u) til G (medmindre den allerede findes).



Incidensmatrix

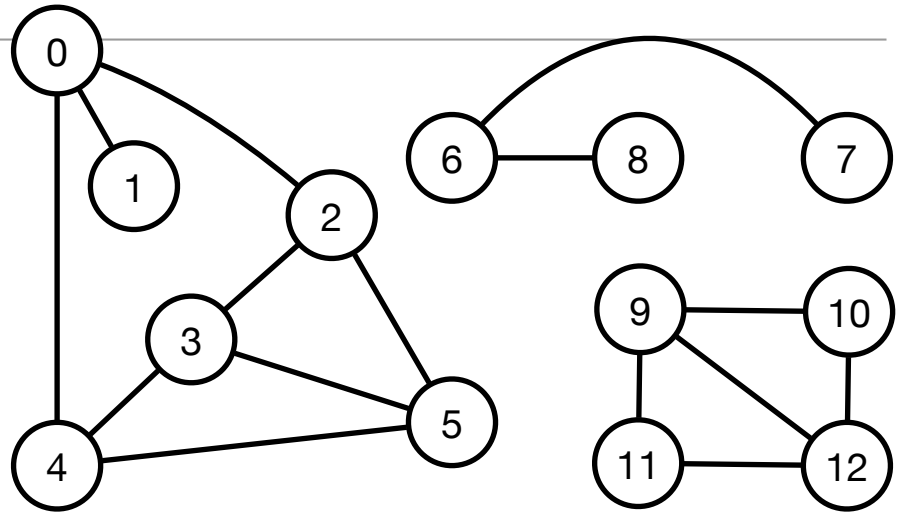
- Graf G med n knuder og m kanter.
- Incidensmatrix (*adjacency matrix*).
  - 2D n × n tabel A.
  - A[i,j] = 1 hvis i og j er naboer og 0 ellers.
- Plads. $O(n^2)$
- Tid.
  - ADJACENT i O(1) tid
  - NEIGHBORS(v) i O(n) tid.
  - INSERT(v, u) i O(1) tid.

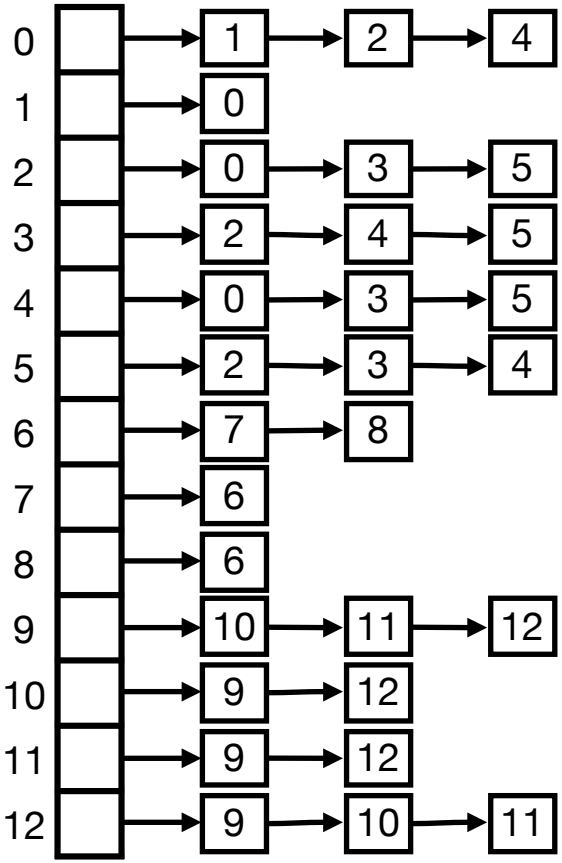


	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	1	0	0	0	0	0	0	0
3	0	0	1	0	1	1	0	0	0	0	0	0	0
4	1	0	0	1	0	1	0	0	0	0	0	0	0
5	0	0	1	1	1	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	1	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0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

Incidensliste

- Graf G med n knuder og m kanter.
- Incidensliste (*adjacency list*).
  - Tabel A[0..n-1].
  - A[i] indeholder liste af naboer af i.
- Plads. $O(n + \sum_{v \in V} \deg(v)) = O(n + m)$
- Tid.
  - ADJACENT, NEIGHBORS OG INSERT i O(deg(v)) tid.





Repræsentation

Datastruktur	ADJACENT	NEIGHBORS	INSERT	Plads
incidensmatrix	O(1)	O(n)	O(1)	$O(n^2)$
incidensliste	O(deg(v))	O(deg(v))	O(deg(v))	O(n+m)

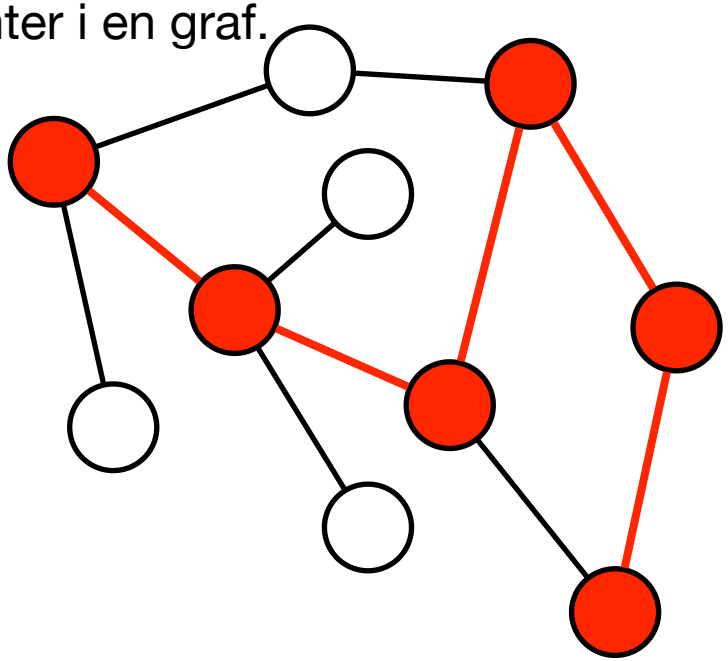
- I den virkelige verden er grafer tynde.

Introduktion til grafer

- Uorienterede grafer
- Repræsentation
- Dybdeførst søgning
  - Sammenhængskomponenter
- Breddeførst søgning
  - Todelte grafer

Dybdeførst søgning

- Algoritme til systematisk at udforske alle knuder og kanter i en graf.
- Dybdeførst søgning *(depth-first search)* fra knude s.
  - Lad alle knuder være umarkede og besøg knude s.
  - Besøg knude v:
    - Marker v.
    - Besøg alle umarkede naboer af v rekursivt.

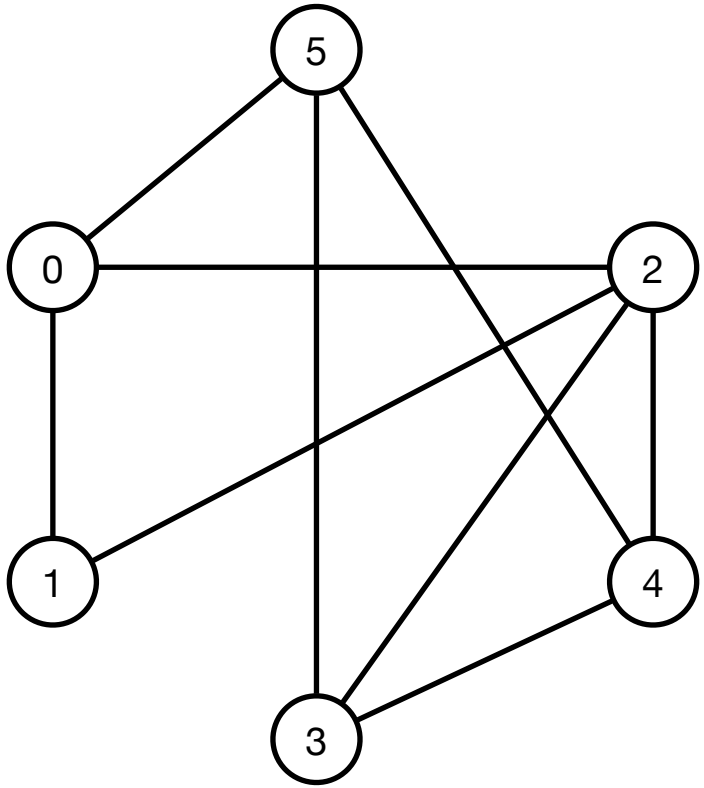


- Intuition.
  - Udforsk ud fra s i en retning indtil vi når "blindgyde".
  - Gå tilbage til sidste knude hvor der var flere uudforskede kanter og fortsæt på samme måde.

- Starttid *(discovery-time)*. Første gang vi besøger en knude i rekursionen.
- Sluttid *(finish-time)*. Sidste gang vi besøger en knude i rekursionen.

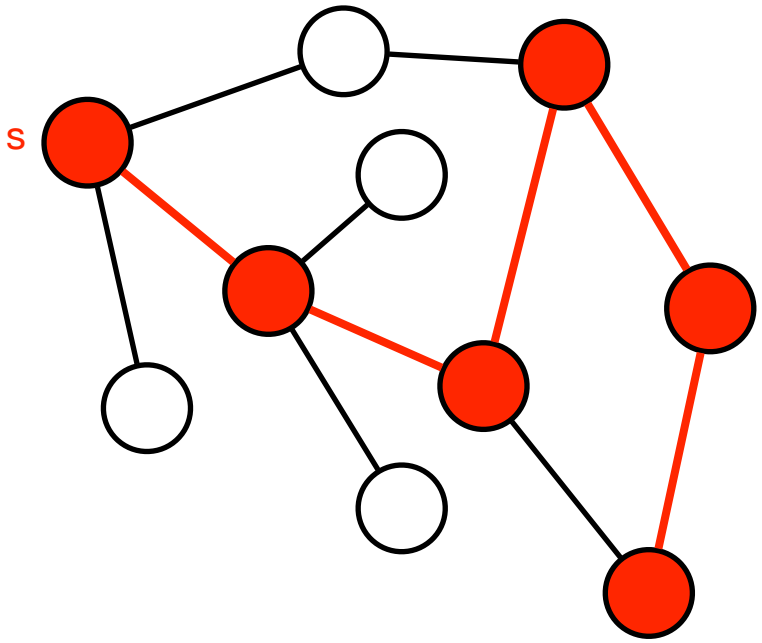
Dybdeførst søgning

- Opgave. Kør DFS fra knude 0 og angiv start- og sluttider. Antag incidenslisterne er sorterede.



Dybdeførst søgning

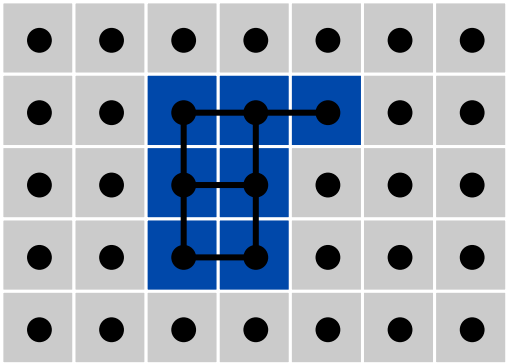
```
DFS(s)
    time = 0
    DFS-VISIT(s)

DFS-VISIT(v)
    v.d = time++
    marker v
    for alle umarkerede naboer u
        DFS-VISIT(u)
        u.π = v
    v.f = time++
```



- Tid. (på incidenslisterepræsentation)
  - Rekursion bliver kaldt på hver knude højst een gang.
  - O(deg(v)) tid ved knude v.
  - $\Longrightarrow$ i alt $O(n + \sum_{v \in V} \deg(v)) = O(n + m)$ tid.
  - Kun knuder forbundet til s bliver besøgt.

Farveudfyldning

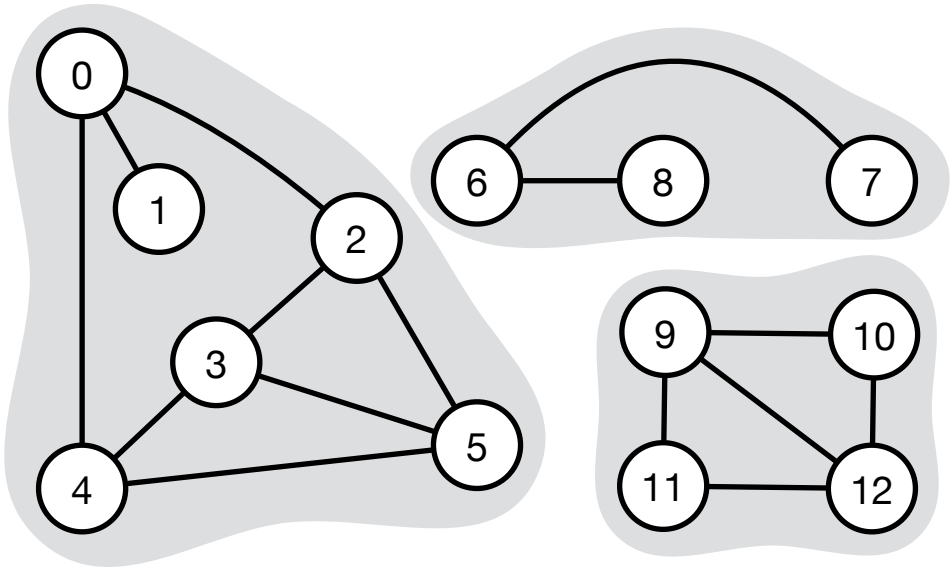
- Farveudfyldning (*flood fill*). Skift farve på sammenhængende område af grønne pixels.



- Algoritme.
  - Byg en gittergraf og kør DFS.
  - Knude: pixel.
  - Kant: mellem nabopixels af samme farve
  - Område: alle knude forbundet til en given knude.

Sammenhængskomponenter

- Def. Et sammenhængskomponent *(connected component)* er en maksimal delmængder af sammenhængende knuder.



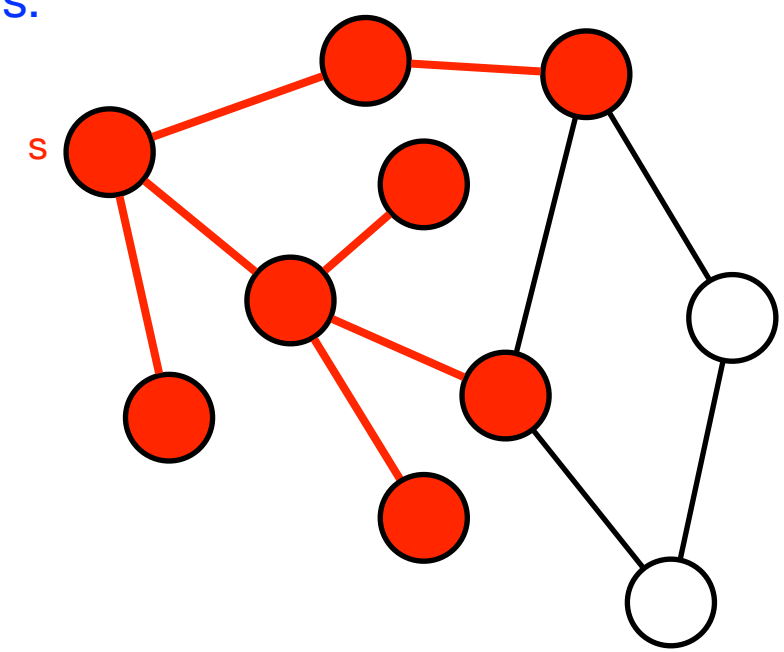
- Hvordan finder vi alle sammenhængskomponenter?
- Algoritme.
  - Lad alle knuder være umarkede.
  - Sålænge der findes en umarkeret knude:
    - Vælg en umarkeret knude v og kør DFS på v.
- Tid. O(n + m).

Introduktion til grafer

- Uorienterede grafer
- Repræsentation
- Dybdeførst søgning
  - Sammenhængskomponenter
- Breddeførst søgning
  - Todelte grafer

Breddeførst søgning

- Breddeførst søgning *(breadth-first search)* fra s.
  - Lad alle knuder være umarkede.
  - Marker s og tilføj s til kø K.
  - Sålænge K ikke er tom:
    - Udtag knude v fra K.
    - For alle umarkede naboer u af v
      - Marker u.
      - Tilføj u til K.

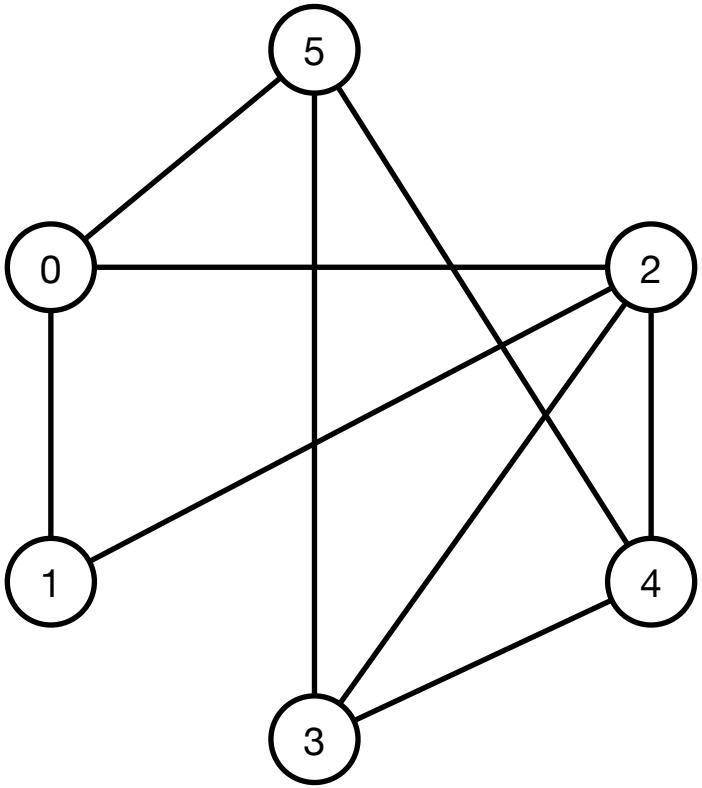


- Intuition.
  - Udforsk ud fra s i alle retninger i stigende afstand fra s.

- Korteste-veje fra s.
  - Afstand fra s i BFS træ = afstand fra s i graf.

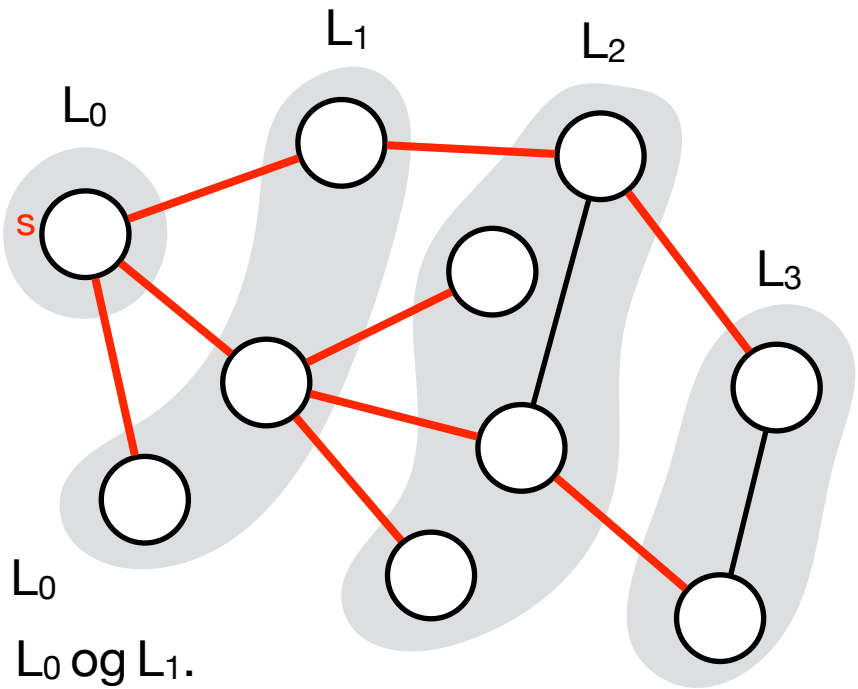
Breddeførst søgning

- Opgave. Kør BFS fra knude 0 og angiv korteste veje. Antag incidenslisterne er sorterede.



Korteste veje

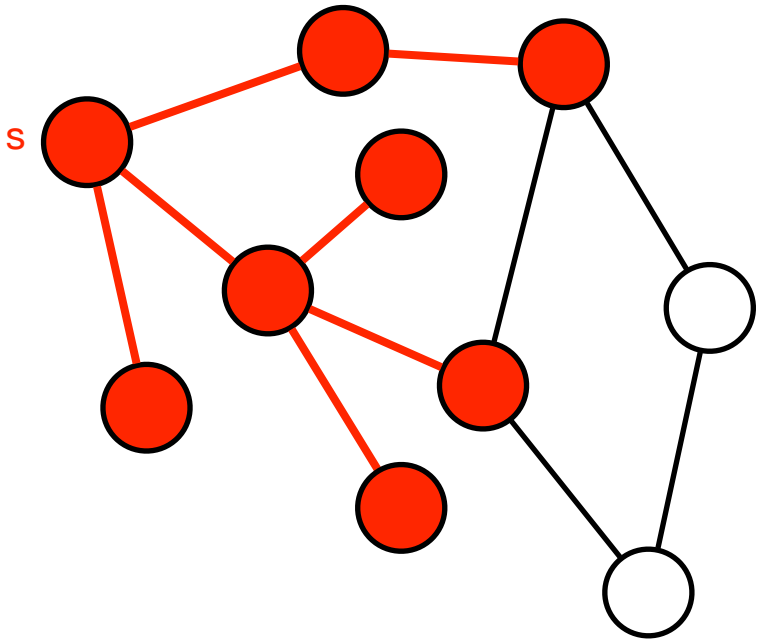
- Lemma. BFS beregner længden af den korteste vej fra s til alle andre knuder.
- Intuition.
  - BFS inddeler knuder i lag *(layers)*. Lag i indeholder knude med afstand i fra s i BFS-træ.



  - Hvad indeholder lagene?
  - $L_0 = \{s\}$
  - L_1 = alle naboer til L_0.
  - L_2 = alle naboer til L_1 der ikke er naboer til L_0
  - L_3 = alle naboer til L_2 der ikke er naboer til L_0 og L_1.
  - ...
  - L_i = alle naboer til L_{i-1} der ikke er naboer til L_j for $j < i-1$
    - = alle knuder med afstand i til s.

Breddeførstsøgning

```
BFS(s)
    marker s
    s.d = 0
    K.ENQUEUE(s)
    gentag indtil K er tom
        v = K.DEQUEUE()
        for alle umarkede naboer u
            marker u
            u.d = v.d + 1
            u.π = v
            K.ENQUEUE(u)
```



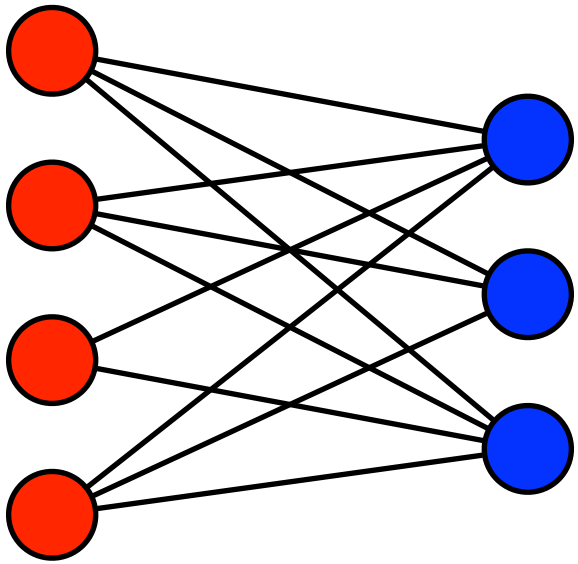
- Tid. (på incidenslisterepræsentation)
  - Knude besøges højst een gang.
  - O(deg(v)) tid ved knude v.
  - $\Longrightarrow$ i alt $O(n + \sum_{v \in V} \deg(v)) = O(n + m)$ tid.
  - Kun knuder forbundet til s bliver besøgt.

Introduktion til grafer

- Uorienterede grafer
- Repræsentation
- Dybdeførst søgning
  - Sammenhængskomponenter
- Breddeførst søgning
  - Todelte grafer

Todelte grafer

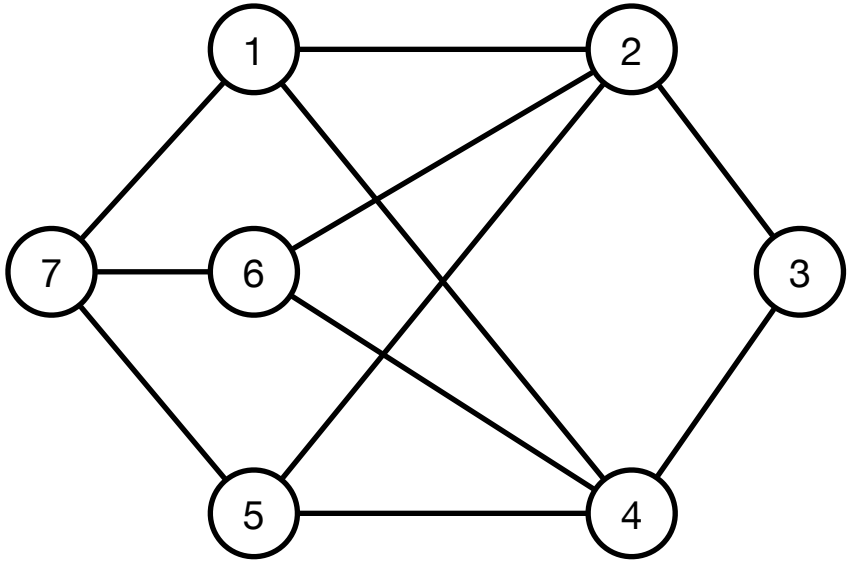
- Def. En graf G er todelt (*bipartite)* hvis og kun hvis alle knuder kan farves røde eller blå således at alle kanter har et rødt og et blåt endepunkt.
- Alternativt def. En graf G er todelt hvis og kun hvis knuder kan deles i to mængder V_1 og V_2 så alle kanter går mellem V_1 og V_2.

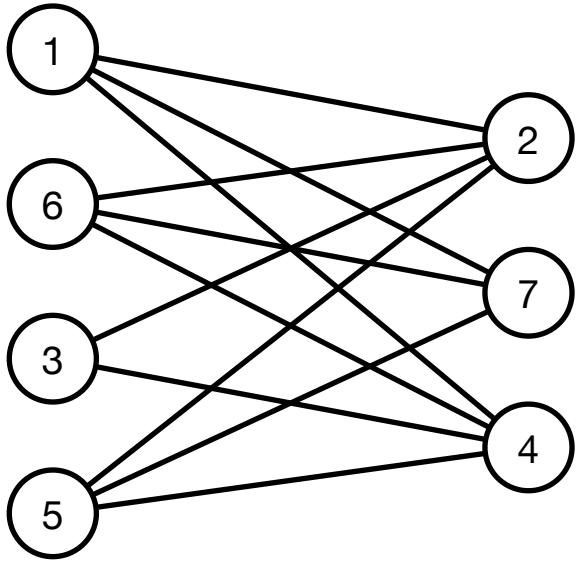


- Anvendelser.
  - Kodning, skedulering, matching, ….
  - Mange grafproblemer er *nemmere* på todelte grafer.

Todelte grafer

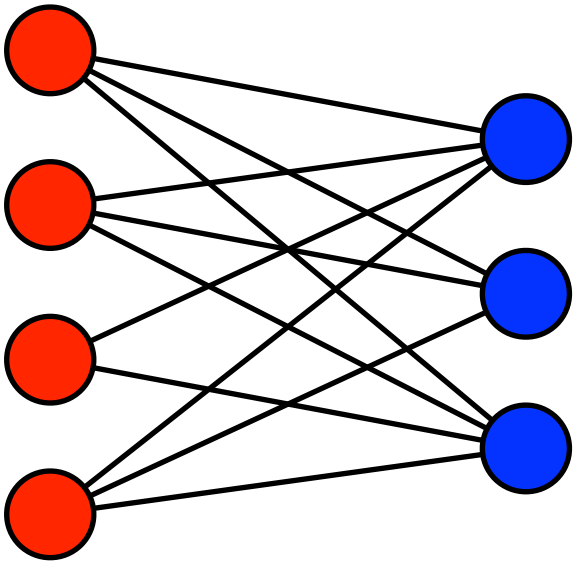
- Udfordring. Givet en graf G, afgør om G er todelt.

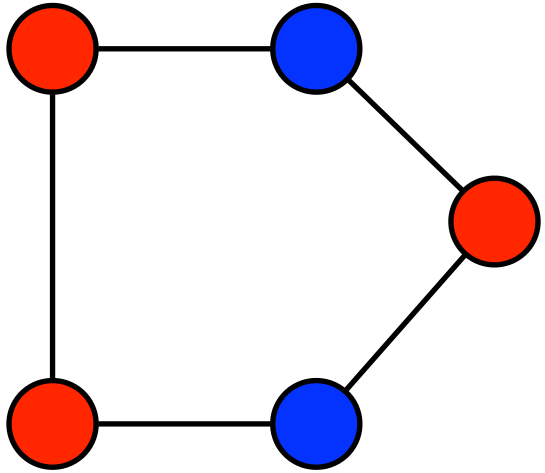




Todelte grafer

- Lemma. En graf G er todelt hvis og kun hvis alle kredse i G har lige længde.
- Bevis.
  - $\Rightarrow$ Hvis todelt så starter og slutter alle kredse i samme side.
  - $\Leftarrow$ Hvis der er kreds af ulige længde kan vi ikke farve den.



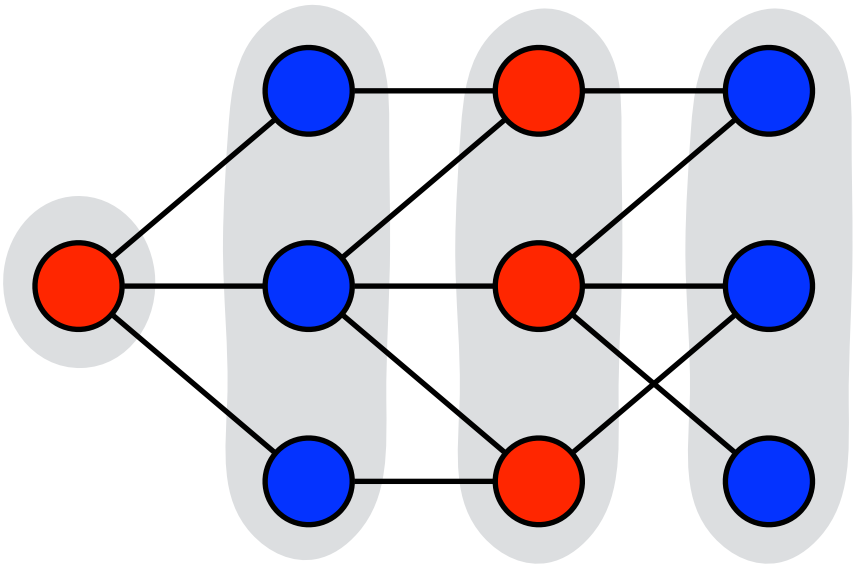


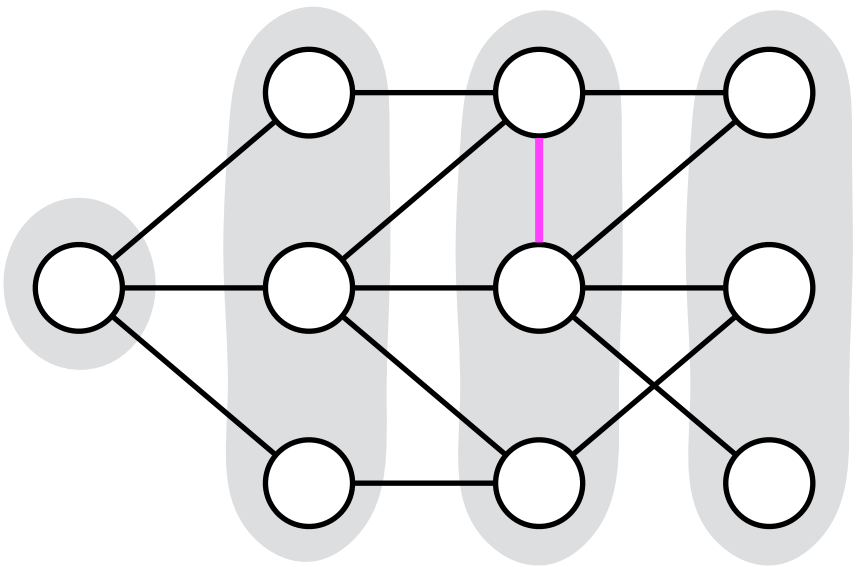
Todelte grafer

- Lemma. Lad G være graf med lag L_0, L_1, …, L_k produceret af BFS. Præcis en af følgende holder:
  1. Ingen kant forbinder 2 knuder i samme lag $\Longrightarrow$ G er todelt.
  2. En kant forbinder 2 knuder i samme lag $\Longrightarrow$ G indeholder en ulige kreds $\Longrightarrow$ G er ikke todelt.
- Bevis.

1. Farv lag skiftevis.

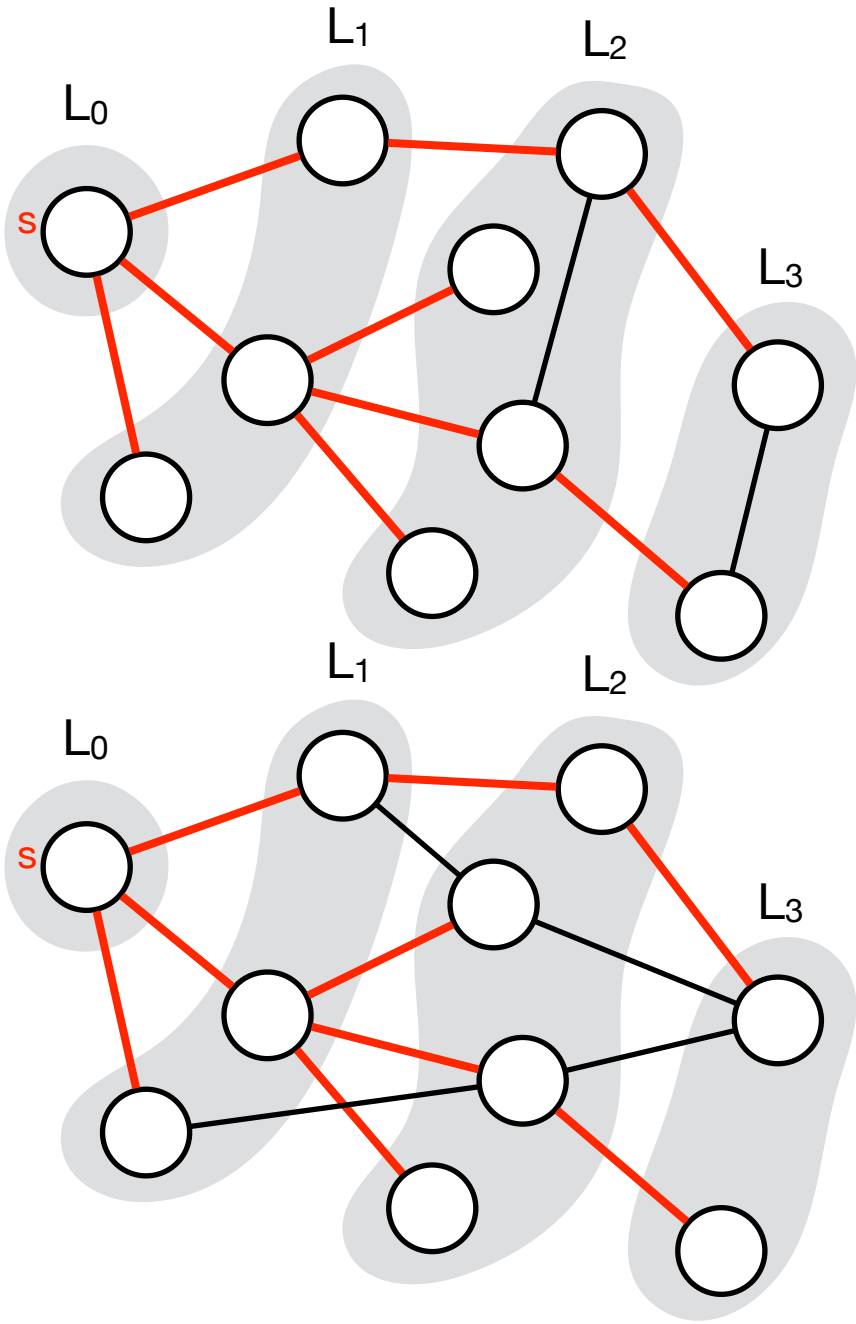
2. Lad (u,v) være kant mellem to knuder i samme lag. Kig på kreds givet ved sti s til u + (u,v) + sti v til s.





Todelte grafer

- Algoritme.
  - Kør BFS på G.
  - For hver kant i G undersøg om den er mellem knuder i samme lag.
- Tid.
  - O(n + m)



Grafalgoritmer

Algoritme	Tid	Plads
Dybdeførst søgning	O(n + m)	O(n + m)
Breddeførst søgning	O(n + m)	O(n + m)
Sammenhængskomponenter	O(n + m)	O(n + m)
Todelte grafer	O(n + m)	O(n + m)

- Antager G er givet i incidenslisterepræsentation

Introduktion til grafer

- Uorienterede grafer
- Repræsentation
- Dybdeførst søgning
  - Sammenhængskomponenter
- Breddeførst søgning
  - Todelte grafer