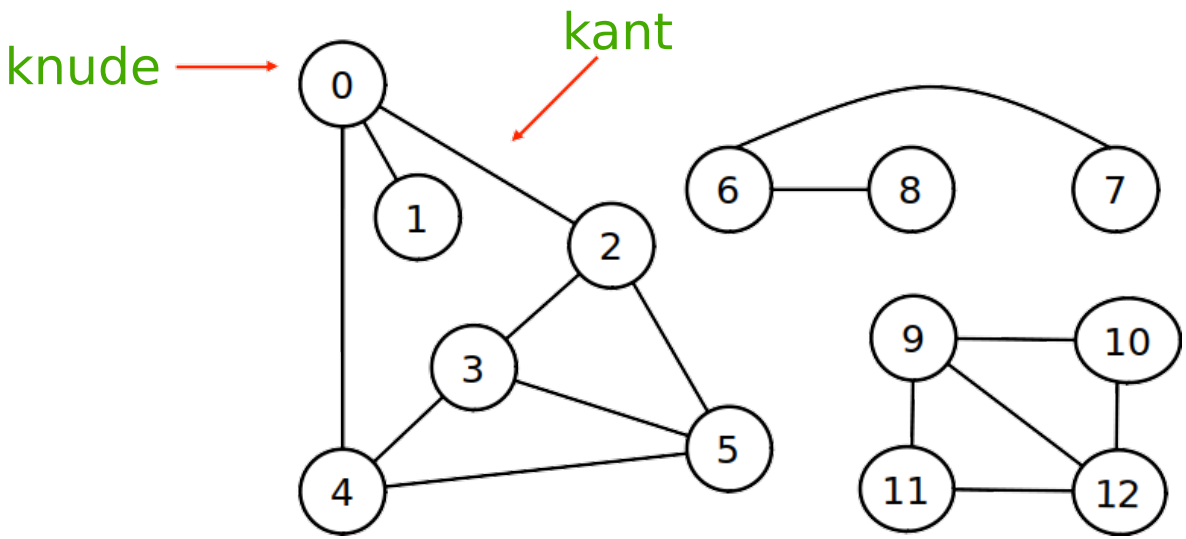
Introduktion til grafer

- Grafer (urettede grafer)
- Repræsentation
- Dybdeførstsøgning (DFS)
  - Sammenhængskomponenter
- Breddeførstsøgning (BFS)
  - Todelte grafer

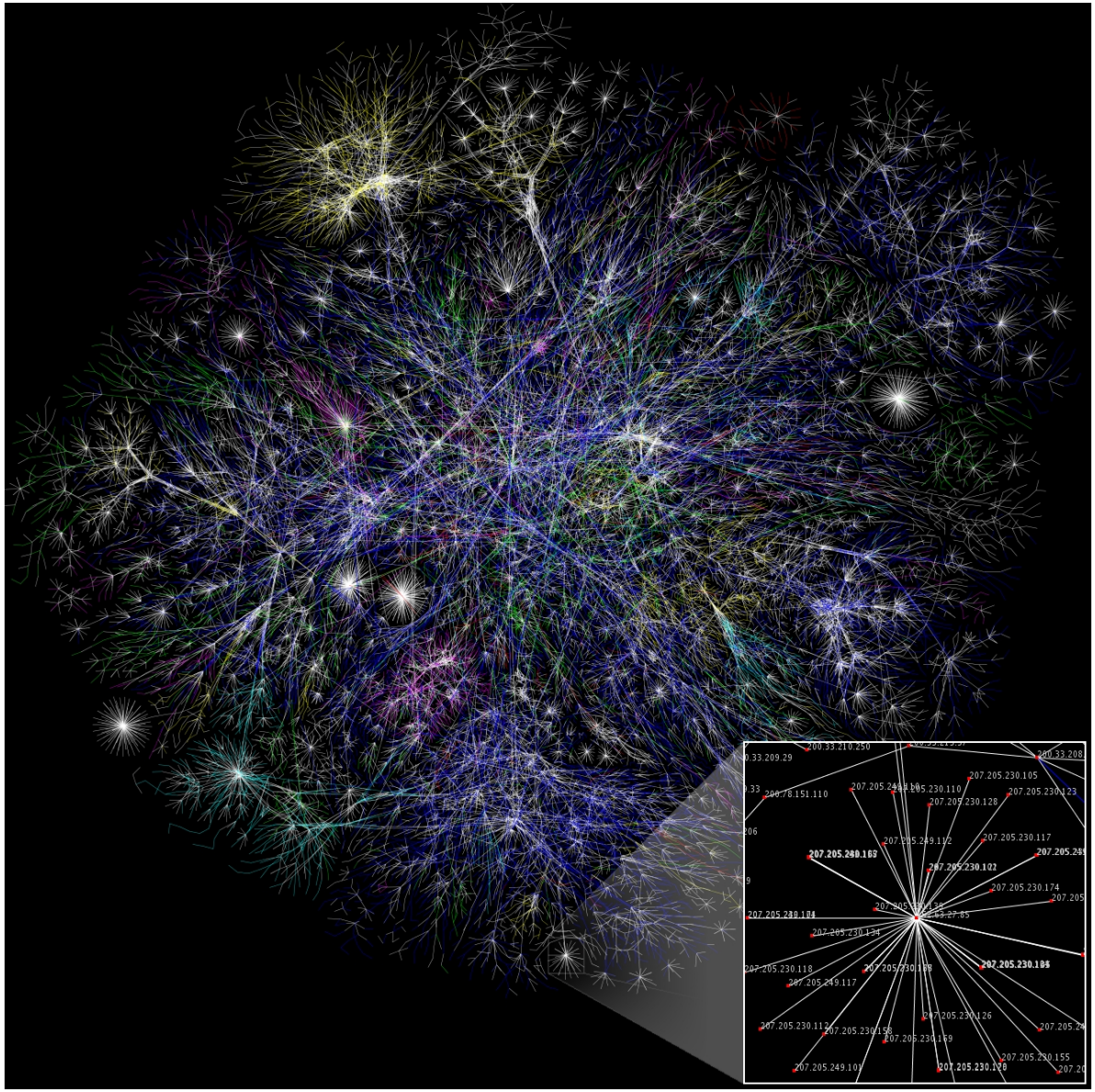
Grafer

- Graf: Mængde af knuder (vertices) parvis forbundet af kanter (edges).



- Grafer
  - Modellerer naturligt mange problemer i mange forskellige områder
  - Tusindvis af praktiske anvendelser
  - Hundredevis af kendte grafalgoritmer

Visualisering af Internettet

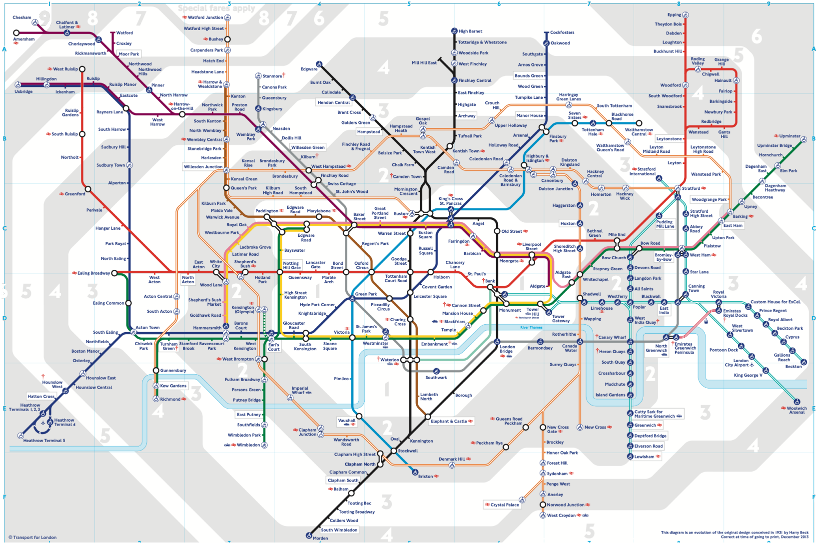


http://en.wikipedia.org/wiki/Opte_Project

Personligt netværk på sociale medier

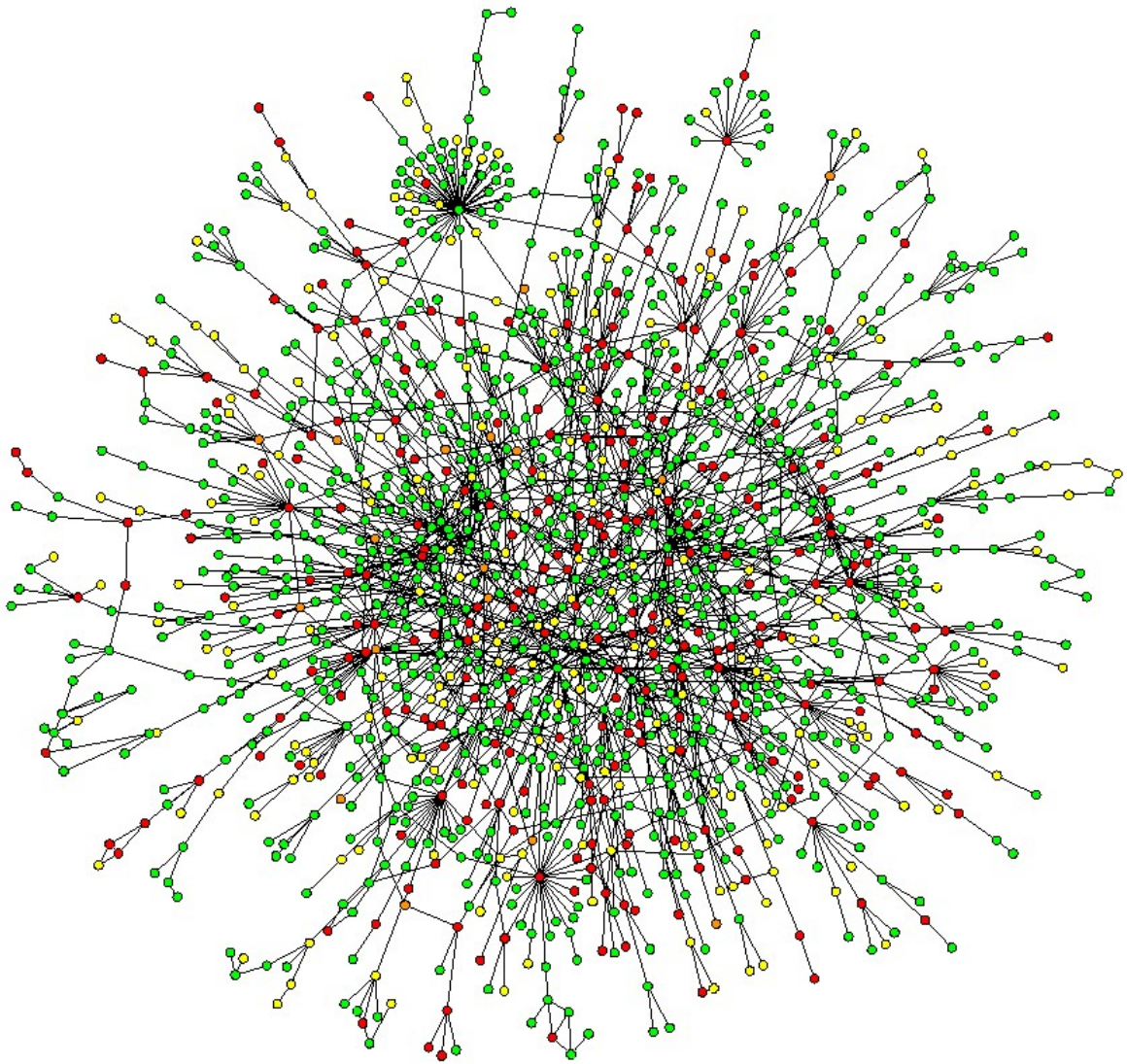
"Visualizing friendships", Paul Butler

Transportnetværk



London metro, London Transport

Proteiners interaktionsnetværk

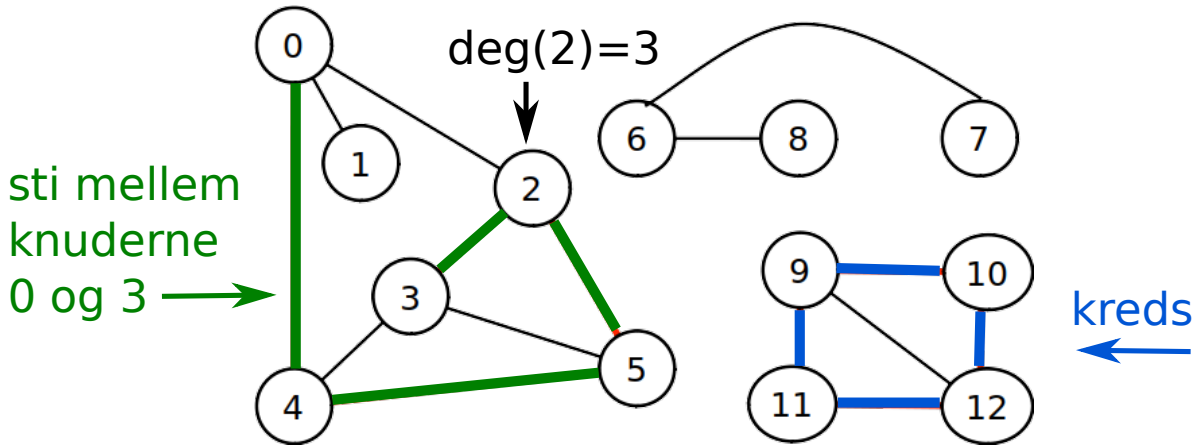


Protein-protein interaktionsnetværk, Jeong et al, Nature Review | Genetics

Tabel over grafanvendelser

graf	knuder	kanter
kommunikation	computer	kabler
transport	vejkryds	veje
transport	lufthavn	flyruter
spil	position	lovligt træk
neuralt netværk	neuron	synapser
finansnetværk	valuta, aktier	transaktioner
kredsløb	logiske porte	forbindelser
fødekæde	dyrearter	rovdyr-byttedyr
molekyle	atom	bindinger

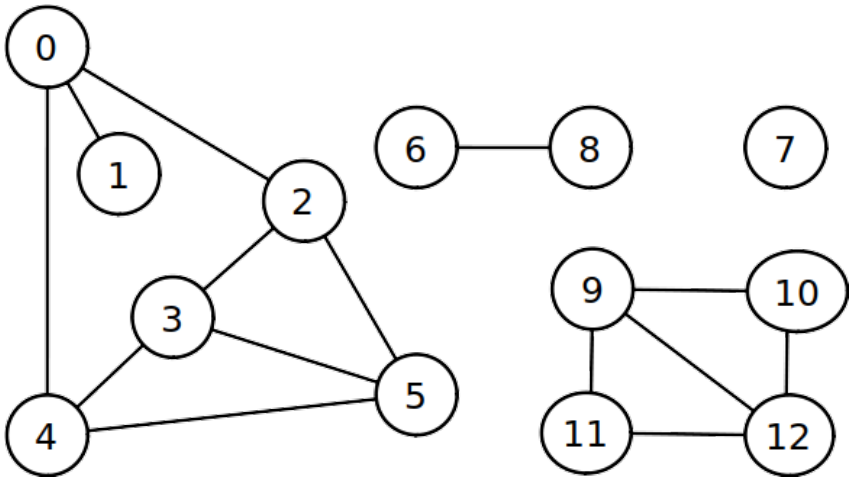
Terminologi



- Graf $G = (V, E)$ (undirected graph, urettet graf).
  - V = mængde af knuder,
  - E = mængde af kanter, (hver kant har 2 endepunkter, disse er naboer)
  - $n = |V|$, $m = |E|$.
- Vandring (walk) : Sekvens af knuder forbundet af kanter.
- Sti ((simple) path) : Vandring, som ikke besøger samme knude flere gange.
- Kreds (cycle) : Cyklisk vandring startende og sluttende i samme knude, som ellers ikke besøger samme knude flere gange.
- Grad (degree) : deg(v) antallet af kanter, hvis ene endepunkt er knuden v.
- Sammenhæng (connectivity) : To knuder er forbundne hvis der er en vej mellem dem. Grafen er sammenhængende hvis alle knuder er forbundne.

Lille lemma

- Husk: $m = |E|$ = antal kanter i grafen.
- Lemma: $\sum_{v \in V} \deg(v) = 2 \cdot m$
- Bevis: Hver kant har 2 endepunkter, og tælles derfor med i summen netop 2 gange.

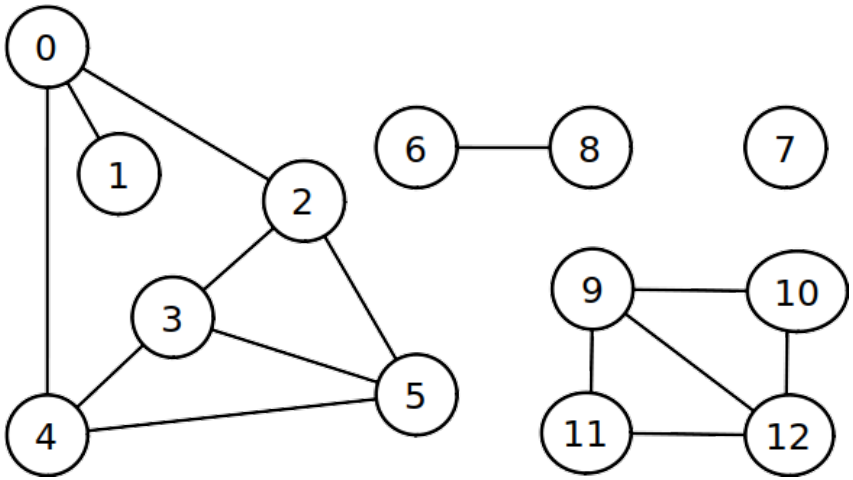


Algoritmiske problemer på grafer

- Vej. Findes der en sti mellem knuderne a og b?
- Korteste vej. Hvad er den korteste vej mellem a og b?
- Længste vej. Hvad er den længste vej mellem a og b?
- Kreds. Er der en kreds i grafen?
- Eulertur. Er der en cyklisk vandring, der benytter hver kant én gang?
- Hamiltonkreds. Er der en kreds, der benytter hver knude én gang?
- Sammenhæng. Er alle knuder forbundne?
- Mindste udspændende træ. Hvad er den 'billigeste' delgraf, der stadigvæk er forbunden?
- Tosammenhængende. For alle knuder v er $G - v$ er sammenhængende?
- Planaritet. Kan grafen tegnes uden krydsende kanter?
- Isomorfi. Er to grafer ens pånær omnavngivning?

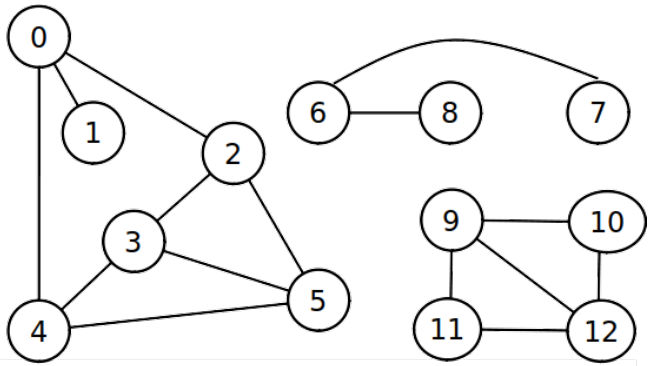
Repræsentation

- En graf G med n knuder og m kanter.
- Repræsentation: Vi skal bruge følgende operationer:
  - adjacent(u, v) – er der en kant mellem u og v?
  - neighbours(v) – returner alle naboer til v.
  - insert(u, v) – indsæt en kant mellem u og v.



Repræsentation ved incidensmatrix

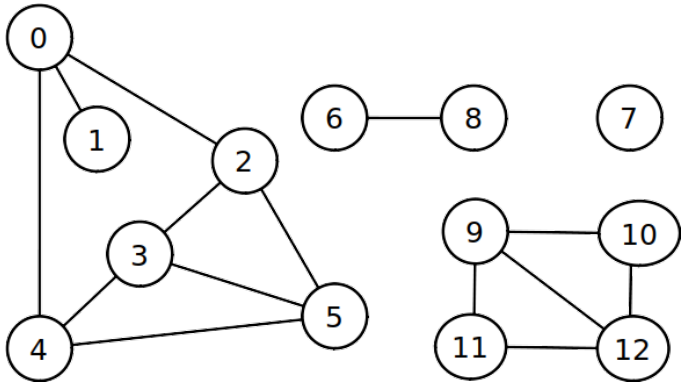
- Graf G med n knuder og m kanter.
  Incidensmatrix (adjacency matrix)
  - $n \times n$ tabel, A
  - $A[i,j] = \begin{cases} 1 & \text{når } i \text{ og } j \text{ er naboer,} \\ 0 & \text{ellers.} \end{cases}$.
- Plads: $\Theta(n^2)$
- Tid:
  - adjacent $\Theta(1)$ tid,
  - neighbours $\Theta(n)$ tid,
  - insert $\Theta(1)$ tid.

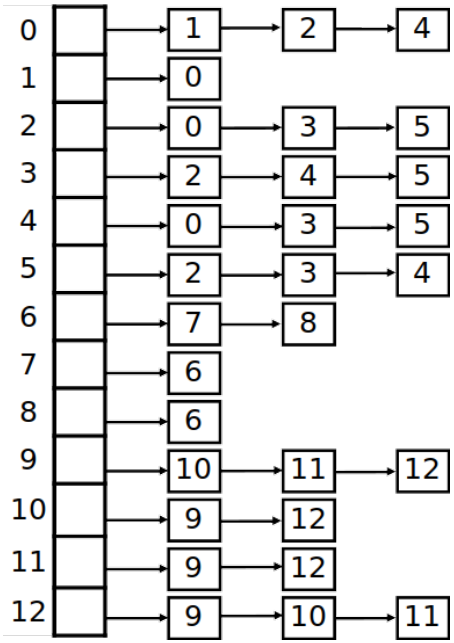


	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	1	0	0	0	0	0	0	0
3	0	0	1	0	1	1	0	0	0	0	0	0	0
4	1	0	0	1	0	1	0	0	0	0	0	0	0
5	0	0	1	1	1	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	1	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0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

Repræsentation ved incidensliste

- Graf G med n knuder og m kanter.
  Incidensliste (adjacency list)
  - Array A af længde n
  - $A[i]$: liste af naboer til i.
- Plads: $\Theta(n + \sum_{v \in V} \deg(v)) = \Theta(n + m)$
- Tid:
  - adjacent(u, v) $\Theta(\deg u)$ tid,
  - neighbours(v) $\Theta(\deg(v))$ tid,
  - insert $\Theta(1)$ tid. Eller $\Theta(\deg(v))$ hvis listen skal være sorteret.





Repræsentationer

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)

- Mange eksempler på grafer er tynde (har få kanter)

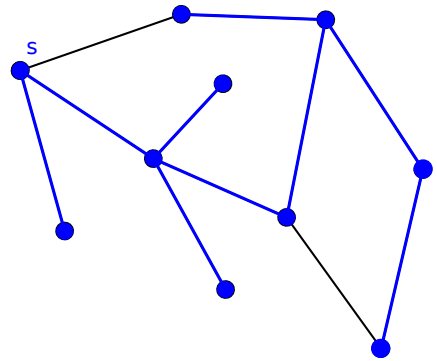
Introduktion til grafer

- Grafer (urettede grafer)
- Repræsentation
- Dybdeførstsøgning (DFS)
  - Sammenhængskomponenter
- Breddeførstsøgning (BFS)
  - Todelte grafer

Dybdeførstsøgning (DFS)

- Algoritme til systematisk at udforske alle knuder og kanter i en graf.

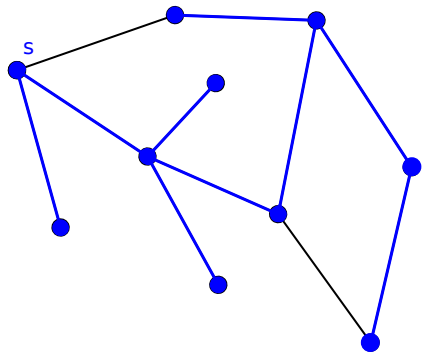
- Dybdeførstsøgning (DFS) fra knuden s:
  - Alle knuder starter som umarkerede.
  - Besøg knude v:
    - Marker v.
    - Besøg rekursivt alle vs umarkerede naboer.



- Opbygger et DFS-træ: kanterne fulgt i rekursive kald.
  - Intuition.
    - Start i s. Udforsk grafen indtil 'blindgyde'.
    - Gå tilbage til sidste knude med uudforskede kanter, og fortsæt på samme måde.
  - Starttid: (discovery time) Første gang, vi besøger knuden.
  - Sluttid: (finish time) Sidste gang, vi besøger knuden.

Dybdeførstsøgning

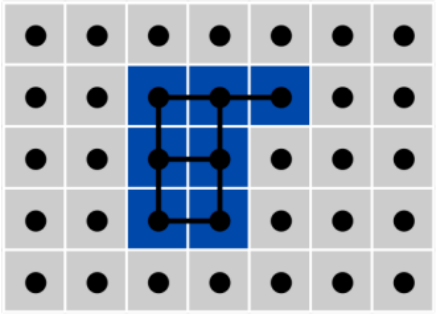
```
DFS(s)
    time = 1
    DFS-VISIT(s)

DFS-VISIT(v)
    v.d = time++ // starttid
       markér v
    for alle umarkerede naboer u
        u.π = v // pointer til forælderen
        DFS-VISIT(u)
    v.f = time++ // sluttid
```



- Tid: (givet incidenslisterepræsentation)
  - Rekursion bliver kaldt på hver knude højest én gang
  - $O(\deg(v))$ arbejde 'skyldes' knude v
  - i alt: $O(n + \sum_{v \in V} \deg(v)) = O(n + m)$ tid.
- Sammenhæng:
  - Besøger præcis de knuder, der er forbundet med s.

Anvendelse: farveudfyldning

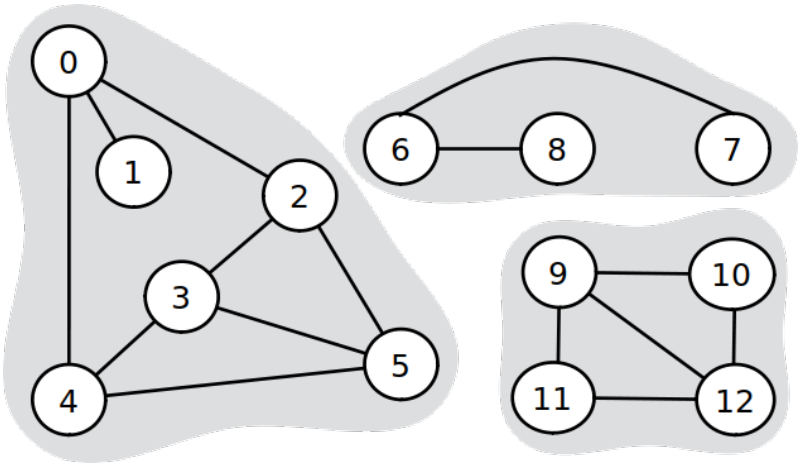
- Farveudfyldning: (flood fill) skift farve på sammenhængende område af ensfarvede pixels.



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

Sammenhængskomponenter

- Sammenhængskomponent: (connected component) maksimal delmængde af sammenhængende knuder.



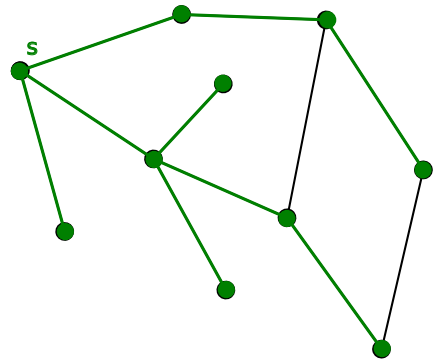
- Hvordan finder vi alle sammenhængskomponenter?
- Algoritme:
  - Alle knuder starter som umarkerede.
  - Så længe der findes en umarkeret knude:
    - kør DFS fra en umarkeret knude.
- Tid: $O(m + n)$.

Introduktion til grafer

- Grafer (urettede grafer)
- Repræsentation
- Dybdeførstsøgning (DFS)
  - Sammenhængskomponenter
- Breddeførstsøgning (BFS)
  - Todelte grafer

Breddeførstsøgning (BFS)

- breddeførstsøgning (BFS) fra knuden s:
  - Alle knuder starter som umarkerede.
  - Marker s og tilføj s til køen Q.
  - Så længe Q ikke er tom,
    - Udtag køens ældste element, knuden v,
    - For alle naboer u til v:
      Hvis u er umarkeret: marker u og føj u til køen Q.
- Opbygger et BFS-træ: kanterne fulgt ved køsættelse.
  - Intuition.
    - Start i s. Udforsk grafen i stigende afstand fra s.
  - Korteste vej: Vej fra i BFS-træ er korteste vej s til v.

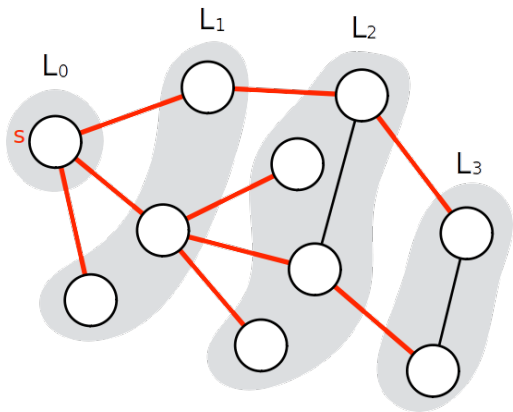


Korteste veje

- Lemma: BFS beregner afstandene mellem s og alle andre knuder.
- Intuition:
  - BFS inddeler grafen i lag.
  - Lag nr. j indeholder knuder med afstand j til s.

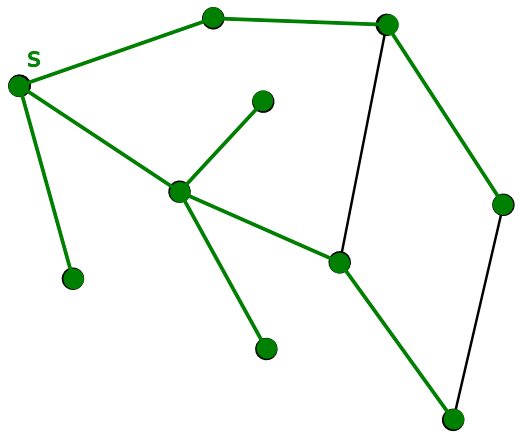
- $L_0 = \{s\}$,
- L_1 = alle naboer til L_0,
- L_2 = alle naboer til L_1, som ikke ligger i $L_0 \cup L_1$,
- L_3 = alle naboer til L_2, som ikke ligger i $L_0 \cup L_1 \cup L_2$,
- ...
- L_j = alle naboer til L_{j-1}, som ikke ligger i noget tidligere lag.

  dvs. alle knuder af afstand j til s.



Breddeførstsøgning

```
BFS(s)
    markér s
    s.d = 0
    Q.ENQUEUE(s)
    gentag indtil Q. ISEMPTY()
        v = Q.DEQUEUE()
        for alle umarkerede naboer u
            markér u
            u.d = v.d + 1
            u.π = v
            Q.ENQUEUE(u)
```



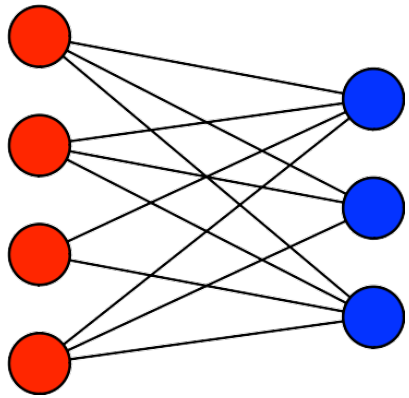
- Tid: (givet incidenslisterepræsentation)
  - Hver knude besøges højst én gang.
  - $O(\deg(v))$ tid på knude v.
  - i alt: $O(n + \sum_{v \in V} \deg(v)) = O(m + n)$ tid.
- Besøger præcis de knuder, der er forbundet med s.

Introduktion til grafer

- Grafer (urettede grafer)
- Repræsentation
- Dybdeførstsøgning (DFS)
  - Sammenhængskomponenter
- Breddeførstsøgning (BFS)
  - Todelte grafer

Todelte grafer

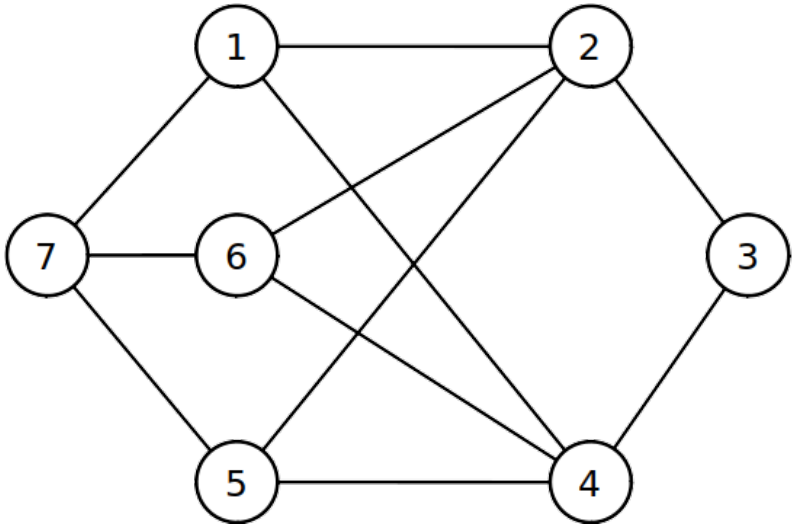
- En graf er todelt (bipartite) hvis dens knuder kan farves rød og blå, så alle kanter har et rødt og et blåt endepunkt.
  Med andre ord:
- En graf er todelt (bipartite) hvis dens knuder kan deles i to mængder V_1 og V_2, så alle kanter går mellem V_1 og V_2.

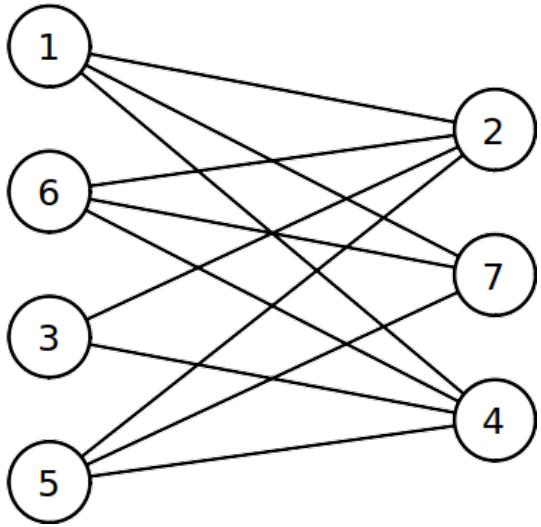


- Matematisk model: fordeling af arbejdsopgaver, tildeling af studiepladser, organer til transplantationspatienter, udbud og efterspørgsel af varer, ...
- Mange grafproblemer er nemmere på todelte grafer.

Todelte grafer

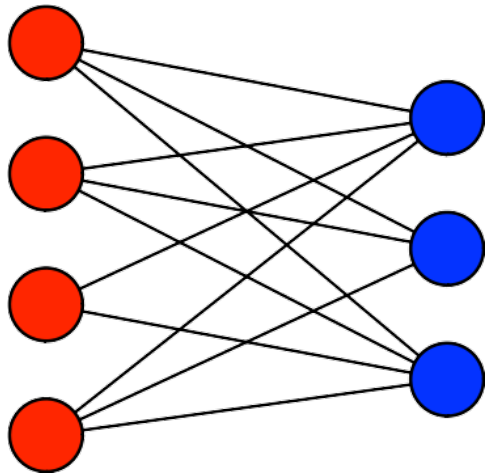
- Udfordring: Givet en graf, afgør, om den er todelt.





Todelte grafer

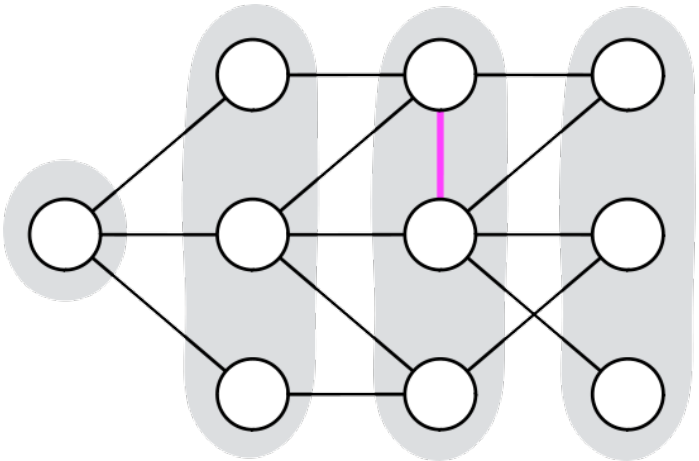
- En graf er todelt hvis og kun hvis den ikke har nogen ulige kredse.
- $\Rightarrow$ Hvis grafen er todelt, så starter og slutter kredsen i samme side.

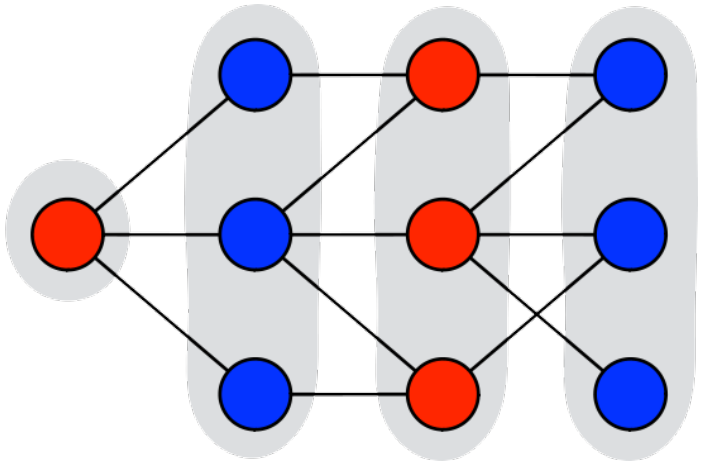


- Omvendt, hvis den har en kreds af ulige længde, kan vi ikke tofarve den: Hvis vi tofarver grafen, tofarver vi også den ulige kreds, men så får to naboer jo samme farve. Med andre ord ... (se næste side)

Todelte grafer

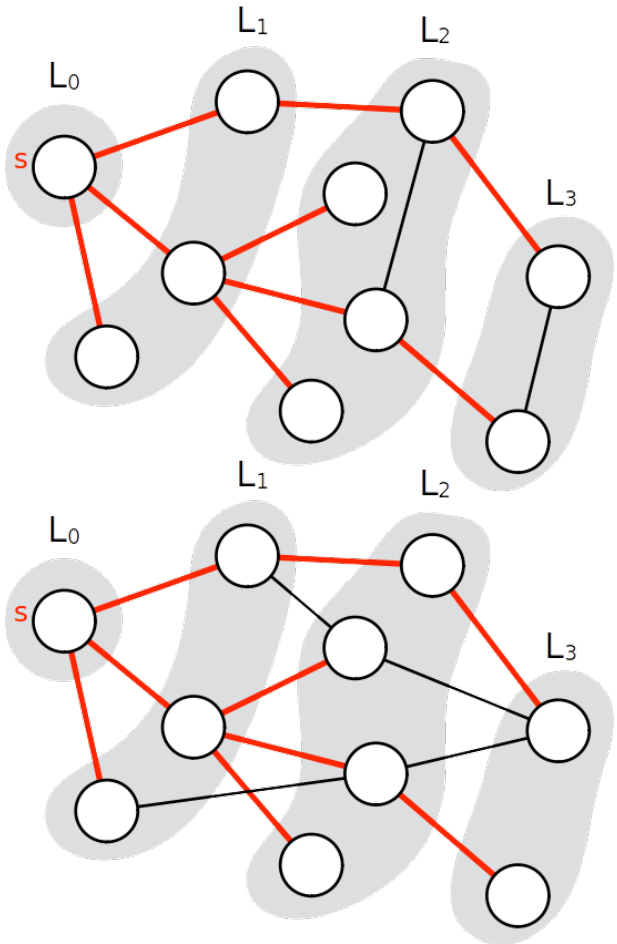
- En graf er todelt hvis og kun hvis den ikke har nogen ulige kredse.
- $\Rightarrow$ Hvis grafen er todelt, så starter og slutter kredsen i samme side.
- $\Leftarrow$ Kig på en knude v og kør BFS fra v.
- Vi får lagene $L_0, L_1, L_2, \ldots$
- Alle kredse har lige længde,
- så derfor er der ingen kanter mellem knuder i samme lag,
- så derfor kan vi farve lagene skiftevis rød og blå,
- så derfor er grafen todelt.





Todelte grafer

- Algoritme:
  - Kør BFS på grafen.
  - For hver kant i grafen, 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)

Introduktion til grafer

- Grafer (urettede grafer)
- Repræsentation
- Dybdeførstsøgning (DFS)
  - Sammenhængskomponenter
- Breddeførstsøgning (BFS)
  - Todelte grafer