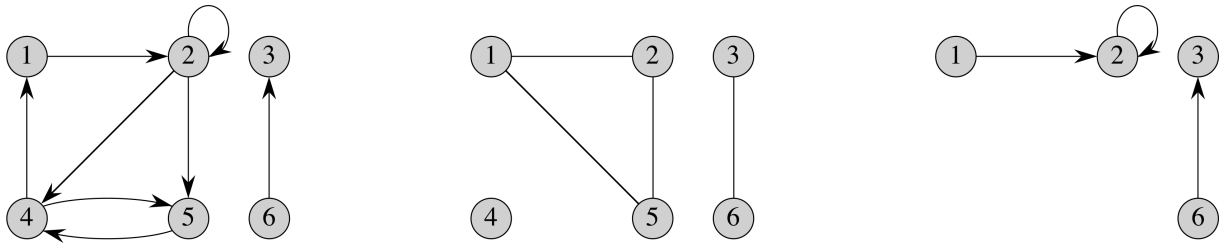
Grafer og graf-gennemløb

Grafer

En mængde V af *knuder* (vertices).

En mængde $E \in V \times V$ af *kanter* (edges). Dvs. ordnede par af knuder.

Figur:

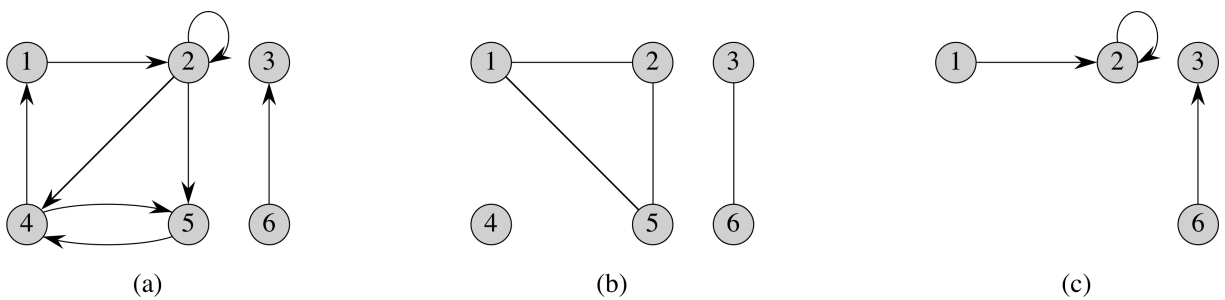


- Terminologi: $n = |V|$, $m = |E|$ (eller V og E (mis)bruges som $|V|$ og $|E|$).
- Orienteret vs. uorienteret.
- $0 \leq m \leq n(n-1)$ og $0 \leq m \leq n(n-1)/2$
- Evt. loops, multiple kanter.
- Vægtede grafer.

Grafer

Modeller for mange ting:

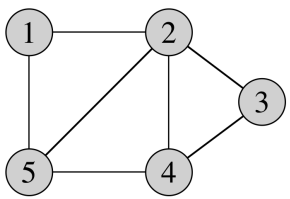
- Ledningsnet (telefon, strøm, olie, vand,...).
- Vejnet.
- Bekendtskaber.
- Medforfatterskaber.
- WWW-grafen af sider og links.



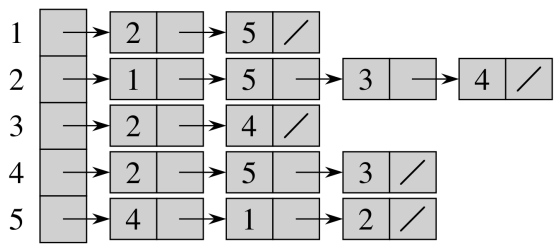
Datastrukturer for grafer

Graf-repræsentationer:

Adjancency list og adjancency matrix



(a)



(b)

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

Plads: Henholdsvis $O(n + m)$ og $O(n^2)$.

Grafgennemløb

Fælles for BFS og DFS:

- Startknude (source) s
- Ikke-mødte knuder: Hvide
- Færdigbehandlede knuder: Sorte
- Front (mødte, men ikke færdigbehandlede): Grå
- Når en knude v opdages første gang: den husker, hvem der opdagede den (dens predecessor) i variablen $v.\pi$. Kanterne $(v, v.\pi)$ udgør et træ med s som rod.
- Vi bruger adjacency list repræsentationen.

Bredde-Først-Søgning (BFS)

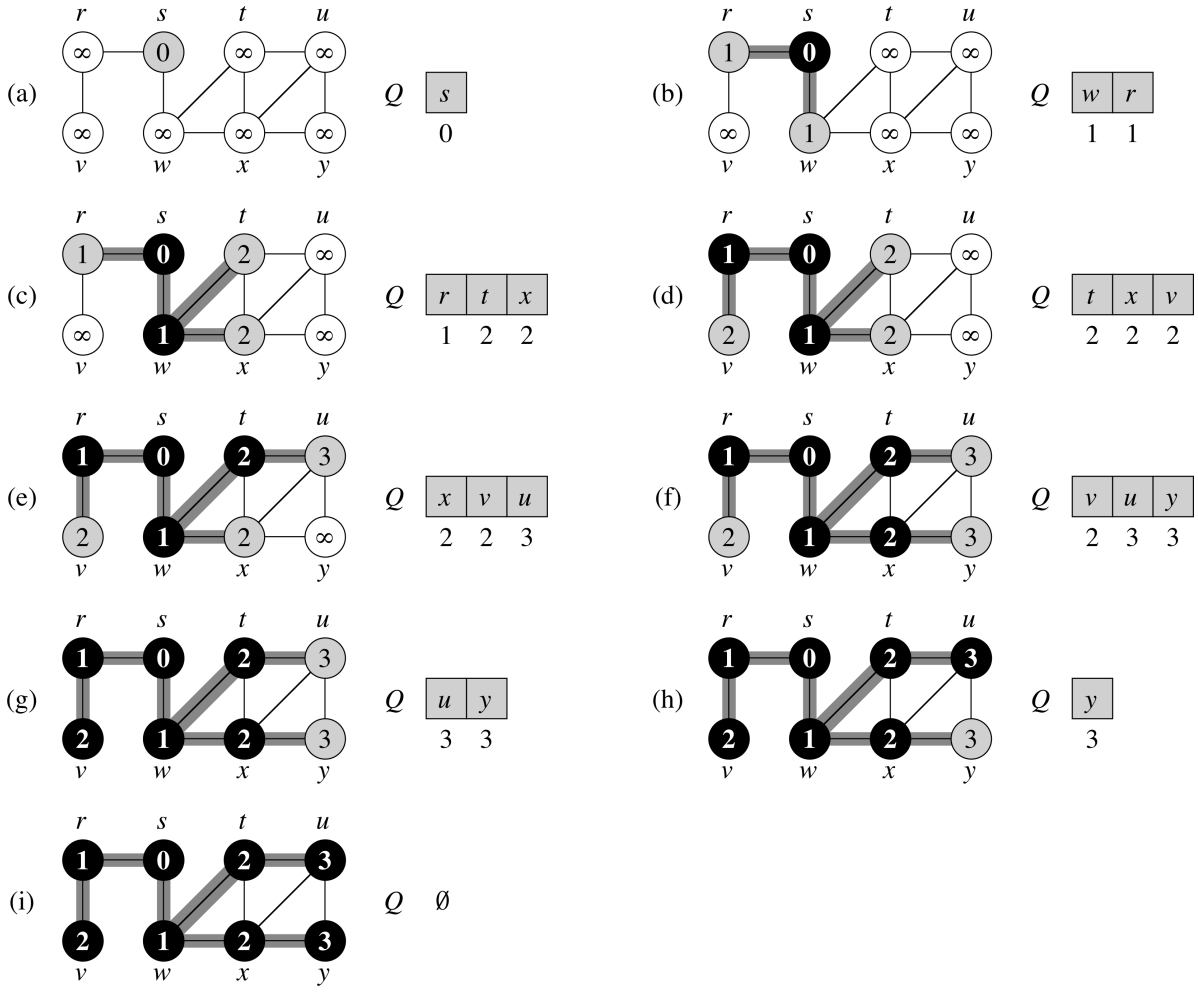
Holder de grå knuder (fronten) i en KØ.

Tilføjer også en distance $v.d$ til alle knuder v.

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2       u.color = WHITE
 3       u.d = ∞
 4       u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11       u = DEQUEUE(Q)
12       for each v ∈ G.Adj[u]
13            if v.color == WHITE
14                 v.color = GRAY
15                 v.d = u.d + 1
16                 v.π = u
17                 ENQUEUE(Q, v)
18       u.color = BLACK
```

Bredde-Først-Søgning (BFS)

Eksempel:



Egenskaber

Køretid: $O(n+m)$

Bevis: Initialisering tager $O(n)$ tid. I resten af algoritmen kan en knude ikke indsættes i køen mere end een gang (pga. farvemærkningen), så maksimalt arbejde er proportionalt med eet gennemløb af alle nabolister, dvs. $O(m)$ arbejde.

Definition: $\delta(s,v)$ er længden af en korteste sti, målt i antal kanter, fra startknuden s til knuden v. Findes ingen sti, defineres $\delta(s,v)=\infty$.

Sætning:

1. Når BFS stopper, har den nået alle knuder v for hvilke der findes en sti fra startknuden s til v.
2. En sti af længde $v.d$ kan for alle nåede knuder findes (i baglæns rækkefølge) ved at følge predecessor-referencer $v.d$ gange, startende fra v (dvs. $v.\pi$, $(v.\pi).\pi$, $((v.\pi).\pi).\pi$,...), og $.d$-værdierne for knuderne på denne sti falder med een for hvert skridt.
3. Når BFS stopper, gælder $v.d=\delta(s,v)$ for alle knuder.

Bevis

Bemærk først at en knude v indsættes i køen een gang, så $v.d$ og $v.\pi$ gives en værdi præcis een gang.

Punkt 1):

Antag der findes en knude, som kan nås med en sti fra s, men som BFS ikke når. Lad v være første knude (set fra s) på denne sti som BFS ikke når, og lad u være knuden før på stien. Da u blev taget ud af køen, burde v (som er i u's naboliste) være nået. Modstrid.

Punkt 2):

Induktion over antal indsættelser i køen: Når en knude u udtages af køen, og en knude v fra u's naboliste indsættes i køen, sættes $v.d = u.d + 1$ og $v.\pi = u$. Hvis udsagnet galdt for u, kommer det klart til at gælde for v. Basis er første indsættelse (af s, for hvilken udsagnet gælder pga. initialiseringen i BFS).

Bevis

Punkt 3):

Definér for heltal $i \leq 0$:

- $\Delta_i = \{v \in V \mid \delta(s, v) = i\}$
- $D_i = \{v \in V \mid v.d = i\}$

Vi viser via induktion på i at for alle i gælder

$$\Delta_i = D_i.$$

Dette medfører punkt 3) for alle knuder, der kan nås fra s. For resten af knuderne forbliver $v.d = \infty$ efter initialisering, så punkt 3) gælder også for dem.

Observér først at BFS-algoritment starter med $D_0 = \{s\}$ i køen, og derefter for $i = 0, 1, 2, 3, \ldots$ udtager alle knuder i D_i mens den indsætter alle knuder i D_{i+1}.

Bevis

Basis (i=0): klar, da $D_0 = \{s\} = \Delta_0$.

Induktionssskridt: antag udsagn er sandt for $i-1$ og lavere, vis det er sandt for i.

For en knude $v \in D_i$ haves fra punkt 2) en sti fra s til v af længde i. Derfor gælder $\delta(s,v) \leq i$. Vi kan ikke have $\delta(s,v) = j < i$, da induktionsantagelse ($\Delta_j = D_j$) så ville give $v.d = j < i$, i modstrid med $v \in D_i$. Derfor er $\delta(s,v) = i$. Dette viser $D_i \subseteq \Delta_i$.

For enhver knude $u \in \Delta_i$ findes en sti fra s til u af længde i. For næstsidste knude w på denne sti gælder $\delta(s,w) = i-1$ (den kan ikke være lavere, for så ville $\delta(s,u)$ være lavere). Fra induktionsantagelsen ($\Delta_{i-1} = D_{i-j}$) får vi at $w.d = i-1$. Da w blev taget ud af køen, var u (en nabo til w) enten hvid, hvorved $u.d$ blev sat til i, eller u var allerede nået fra en naboknude t, som allerede var taget ud og derfor (via observation på sidste side) har $t.d \leq w.d = i-1$. I det sidste tilfælde blev $u.d$ blev sat $\leq i$. Men det gælder af punkt 2) altid at $i = \delta(s,u) \leq u.d$. I begge tilfælde haves $u.d = i$. Dette viser $\Delta_i \subseteq D_i$.

Alt i alt: $\Delta_i = D_i$.

Dybde-Først-Søgning (DFS)

Holder de grå knuder (fronten) i en STAK.

Stakken er implicit i den rekursive formulering nedenfor (er rekursionsstakken), men kan også kodes eksplicit.

(Mere præcist: elementerne på stakken er de grå knuder, hver med en delvist gennemløbet naboliste [gennemløb i for-løkken i DFS-VISIT])

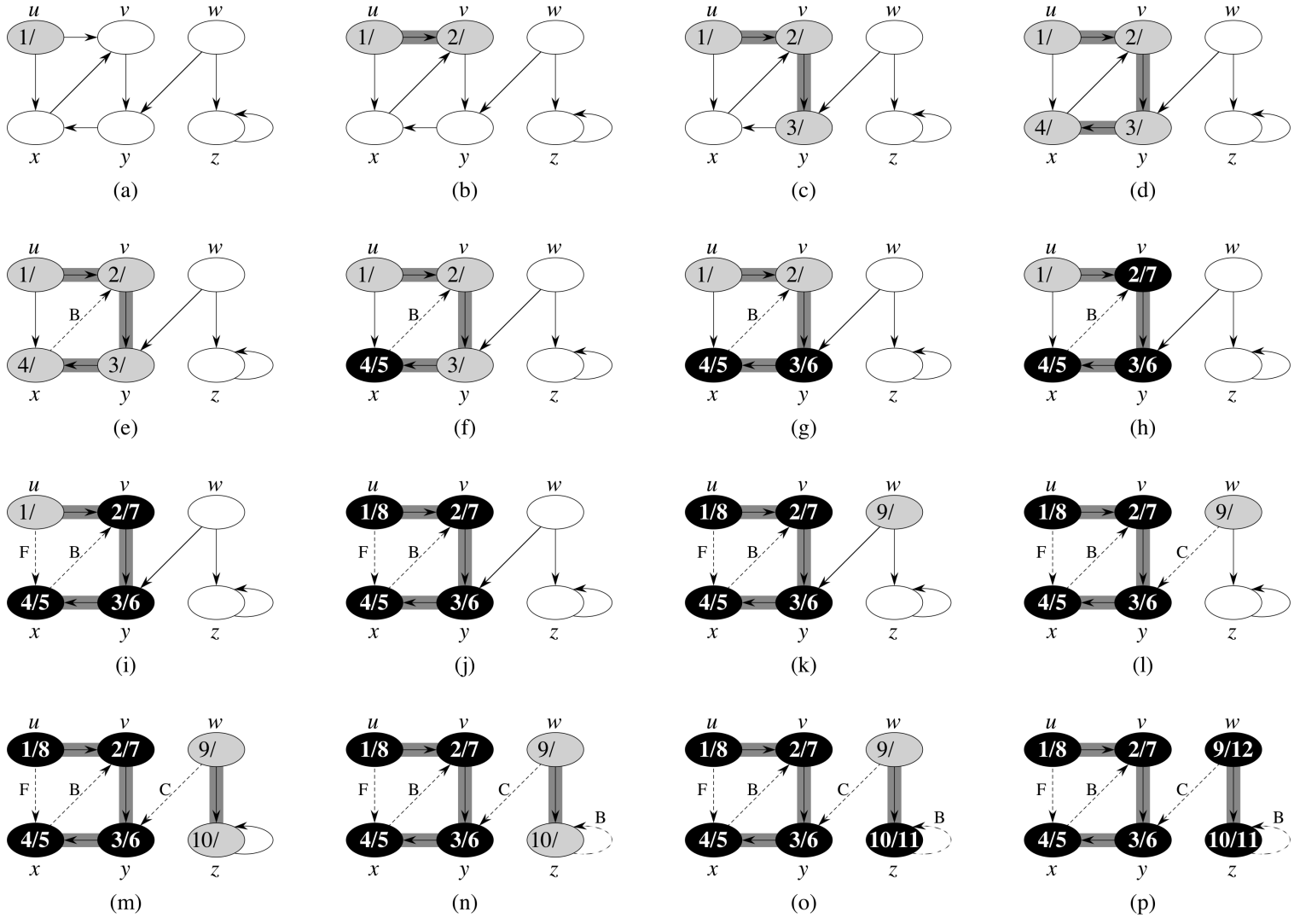
DFS tilføjer også timestamps $u.d$ for "discovery" (hvid $\rightarrow$ grå) og $u.f$ for "finish" (grå $\rightarrow$ sort) til alle knuder u.

```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
1  time = time + 1          // white vertex u has just been discovered
2  u.d = time
3  u.color = GRAY
4  for each v ∈ G.Adj[u]    // explore edge (u, v)
5      if v.color == WHITE
6          v.π = u
7          DFS-VISIT(G, v)
8  u.color = BLACK          // blacken u; it is finished
9  time = time + 1
10 u.f = time
```

Dybde-Først-Søgning (DFS)

Eksempel:



Egenskaber

Køretid: $O(n + m)$

Bevis: Den ydre algoritme DFS tager $O(n)$ tid. Hovedalgoritmen $\mathrm{DFS\text{-}Visit}$ kan ikke kaldes på en knude mere end een gang (pga. farvemærkningen), så arbejdet dér er proportionalt med eet gennemløb af alle nabolister, dvs. $O(m)$ arbejde.

Observér:

- Discovery (hvid $\rightarrow$ grå) af v = kald af $\mathrm{DFS\text{-}Visit}$ på v = PUSH af v på stakken.
- Finish (grå $\rightarrow$ sort) af v = retur fra kald af $\mathrm{DFS\text{-}Visit}$ på v = POP af v fra stakken.

Heraf:

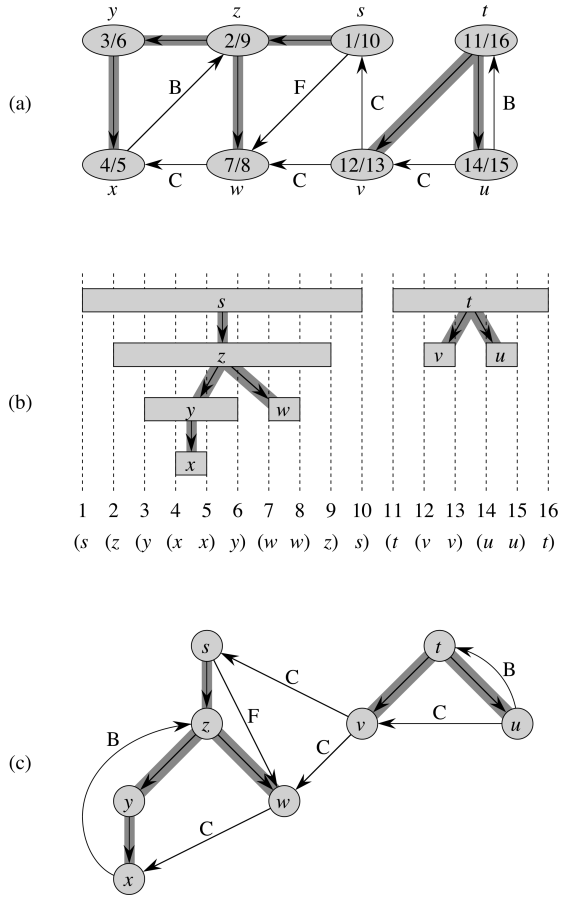
- Kanterne $(v, v.\pi)$ er præcis rekursionstræerne for $\mathrm{DFS\text{-}Visit}$ (eet træ for hvert kald fra DFS).
- Intervallet $[v.d, v.f]$ er den periode v er på stakken.

Egenskaber

Hvis u og v er på en stak samtidig, og v er pushed sidst, må v poppes før u kan poppes.

Heraf følger at for alle knuder u og v må intervallerne $[u.d, u.f]$ og $[v.d, v.f]$ enten være disjunkte (u og v var ikke på stakken samtidig) eller det ene må være helt indeholdt i den andet.

(Discovery- og finish-tider er derfor nestede som parenteser er det.)



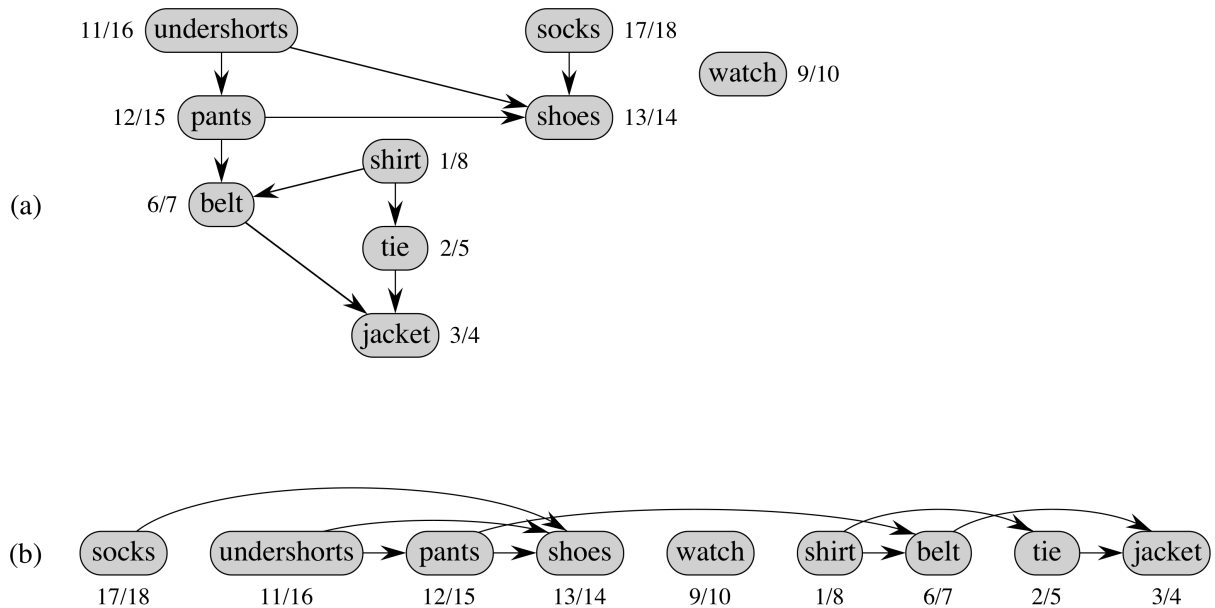
Kanter (u, v) (undersøgt fra u) kan deles i: *tree-kanter* [v hvid], *back-kanter* [v ikke-hvid (grå) og v's interval indeholder u's], *forward-kanter* [v ikke-hvid (sort) og v's interval er indeholdt i u's] og *cross-kanter* [v ikke-hvid (sort) og v's interval er før u's].

For uorienterede er der kun tree-kanter og back-kanter (hvis en kants type bestemmes første gang den undersøges fra een af dens ender).

DAGs og topologisk sortering

DAG = Directed Acyclic Graph.

Topologisk sortering af en DAG: en lineær ordning af knuderne så alle kanter går fra venstre til højre.



DAGs og topologisk sortering

Lemma: En orienteret graf har en kreds (cycle) $\Leftrightarrow$ der findes back-edges under et DFS-gennemløb.

Bevis:

$\Rightarrow$: Se på første knude v i kredsen som bliver grå - dvs. alle andre knuder u i kredsen har $v.d < u.d$. Da intervaller enten er helt indeholdt i hinanden eller er disjunkte, gælder enten $v.d < u.d < u.f < v.d$ eller $v.f < u.d$.

Antag det sidste tilfælde forekommer, og se på den første (set fra v) sådanne knude u i kredsen, og lad w være dens forgænger (evt. v selv). Da haves $w.f \leq v.f < u.d$, men pga. kanten (w, u) kan dette ikke forekomme (kanten må blive undersøgt, og u opdaget inden w er færdig).

Så kun første tilfælde forekommer. Specielt gælder dette sidste knude u' i kredsen (som peger på v), hvorved kanten (u', v) erklæres en backedge.

$\Leftarrow$: Når en back-edge findes: Der er en kreds af trækanter (mellem knuderne som lige nu er på stakken) og een back-kant.

DAGs og topologisk sortering

Lemma: For en kant (u, v) gælder $u.f < v.f \Leftrightarrow$ kanten er en back-edge.

Bevis: Check de fire edge cases (tree, back, forward, cross), brug parentes-strukturen af discovery- og finish-tider.

Korollar til to foregående lemmaer: Graf er en DAG $\Leftrightarrow$ DFS finder ingen back-edges $\Leftrightarrow$ ordning af knuder efter faldende finish-tider giver en topologisk sortering.

```
TOPOLOGICAL-SORT(G)
1  call DFS(G) to compute finishing times v.f for each vertex v
2  as each vertex is finished, insert it onto the front of a linked list
3  return the linked list of vertices
```

Tid: $O(n + m)$.