

# IN2010 - Gruppe 5

Uke 6 - Grafer

Hva er grafer?

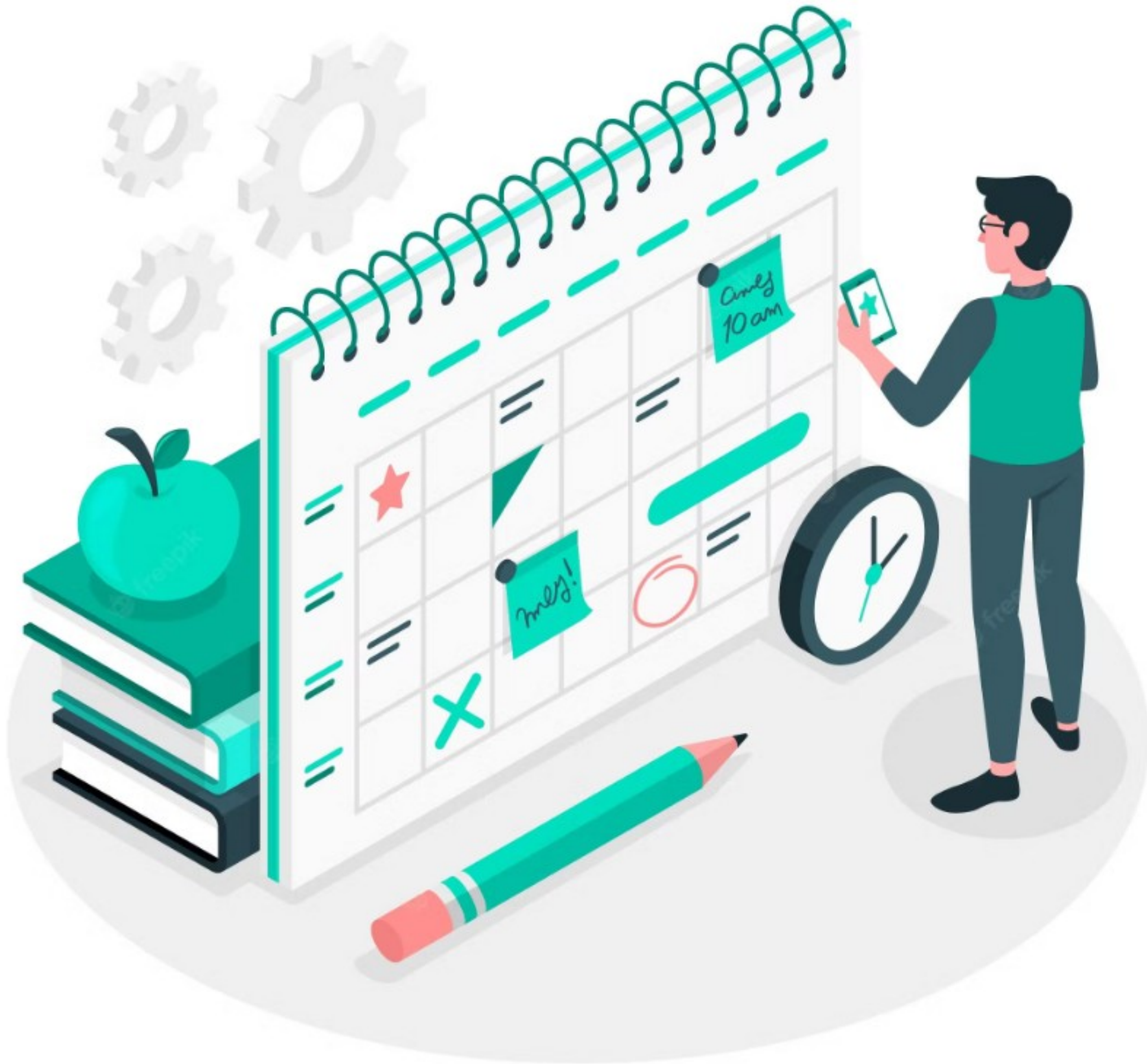
Hvordan representeres de?

Hvordan traverseres grafer?

Hvordan kan de ordnes?



# Bli med!



# Dagens Plan

- Oblig 2 gjennomgang
- Pensum-gjennomgang
- Live koding
- Gruppeoppgaver

# Oblig 2 er ute!

<https://www.uio.no/studier/emner/matnat/ifi/IN2010/h22/Innleveringer/innleveringsoppgave2/innleveringsoppgave2.pdf>

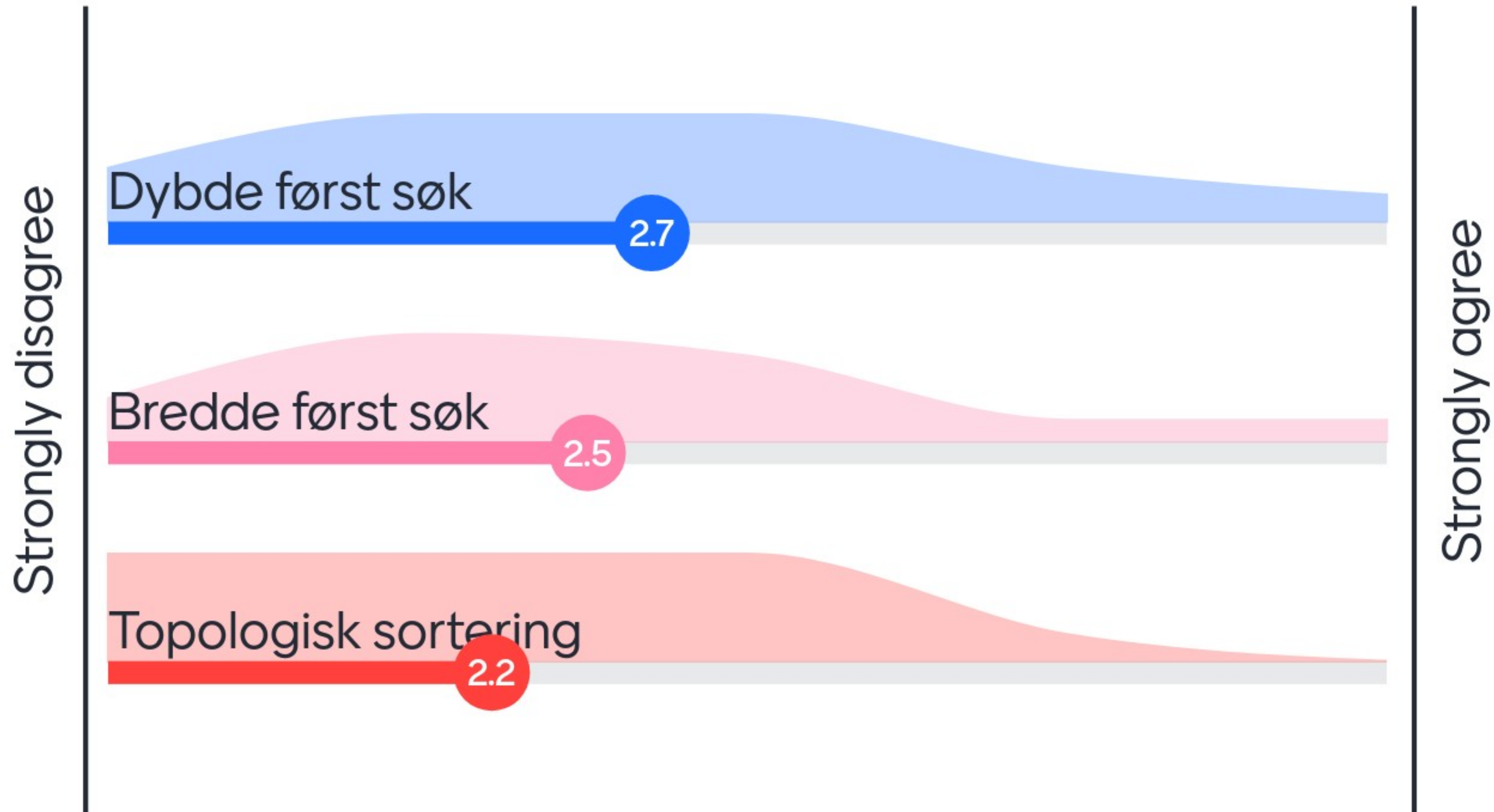
GitHub  
Input

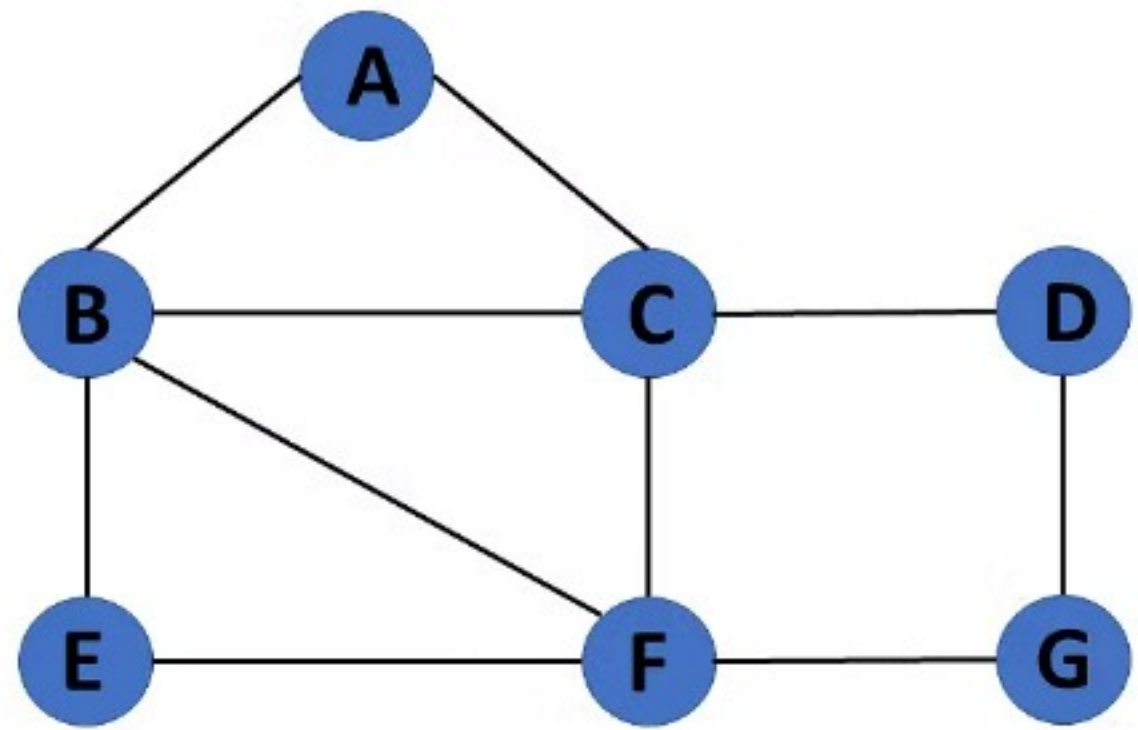


# Pensum-gjennomgang



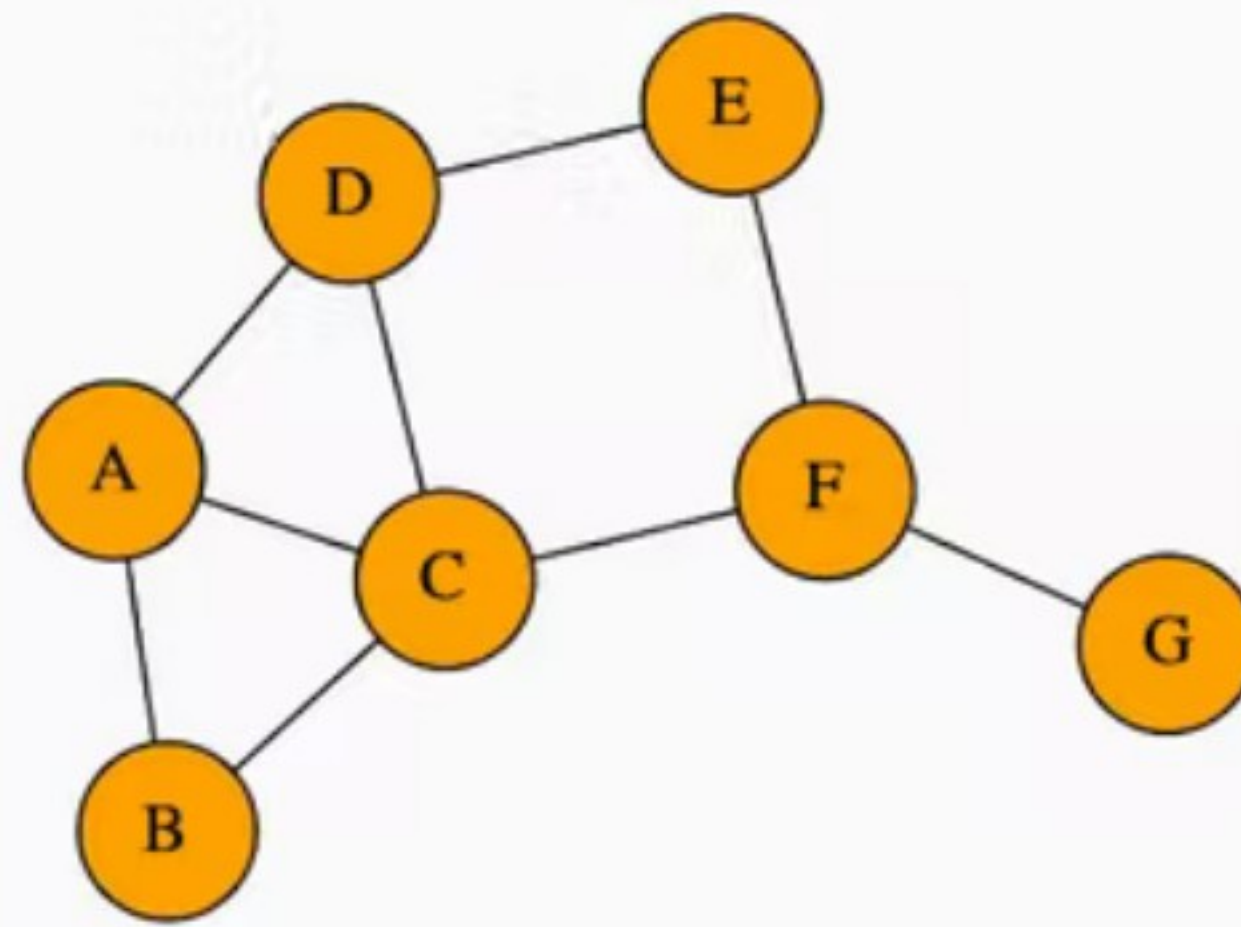
# Hvor godt forsto du ukens pensum?





# Grafer Basics

- Hva er en graf?
- Hvordan Representerer vi grafer?



- noder  $\{A, B, C, D, E, F, G\}$
- Kanter  $\{A, B\}, \{B, C\}, \{C, F\}, \dots$

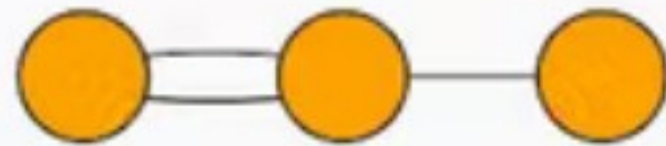

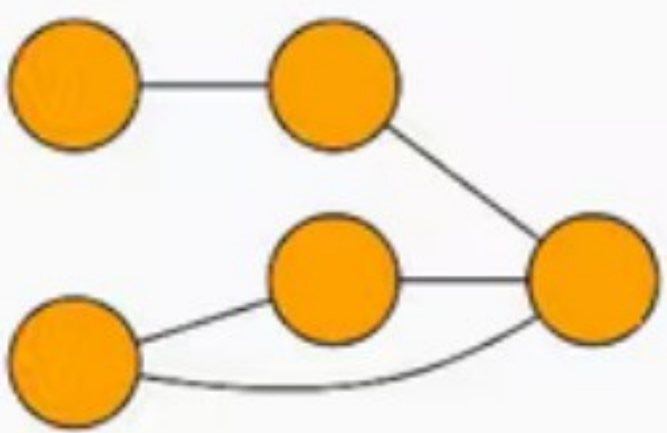
Formelt består en graf av to mengder  $(V, E)$

- $V$  en mengde noder,
- $E$  en mengde kanter mellom noder fra  $V$ .  $\{u, v\}$  i  $E$  representerer at det finnes en kant mellom  $u$  og  $v$ .

Grafer







Betegnelse	Forklaring	Eksempel
Vektet/uvektet	Kantene har en verdi/kostnad knyttet til seg	
Parellelle kanter	Flere enn én kant mellom to noder	
(Enkle) løkker	En kant fra en node til seg selv	
Enkel graf	Ingen løkker, ingen parallelle kanter, urettet, uvektet	

## Terminologi

# Rettete/Urettede grafer

En kant i en graf kan ha en retning.  vs 

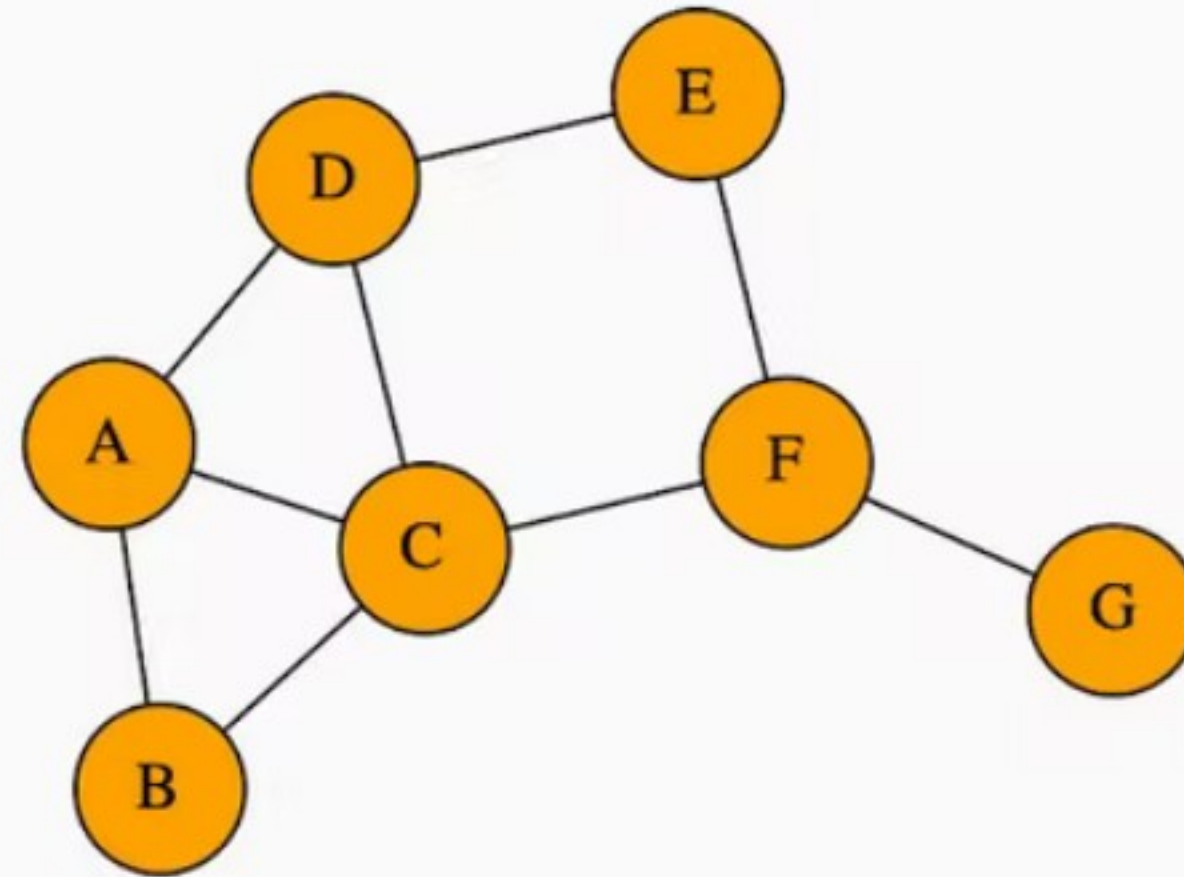
- For urettede grafer, der kanter ikke har en retning, skriver vi  $\{u, v\}$  for en kant mellom  $u$  og  $v$
- For rettede grafer, der kanter har en retning, skriver vi  $(u, v)$
- urettede grafer kan bli representert som rettede grafer:  $\{u, v\}$  blir til  $(u, v)$  og  $(v, u)$ .

Rettete/Urettede grafer





## Veier og stier



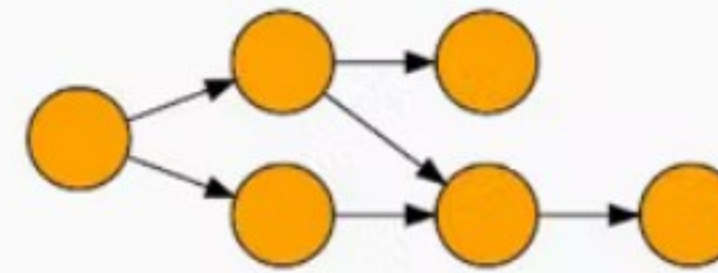
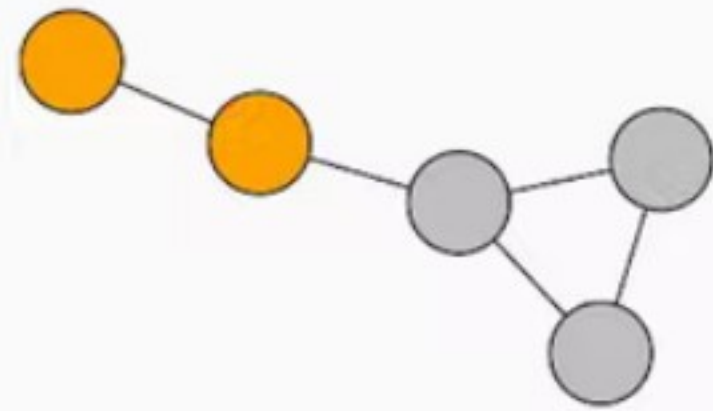
- Sti: En sekvens av noder i grafen forbundet av kanter, der ingen *noder* gjentas. A, B, C, D
- Vei: En sekvens av noder i grafen forbundet av kanter, der ingen *kanter* gjentas. A, B, C, A, D

Hvordan finne den korteste stien fra en node til alle andre?  $\rightsquigarrow$  neste uke

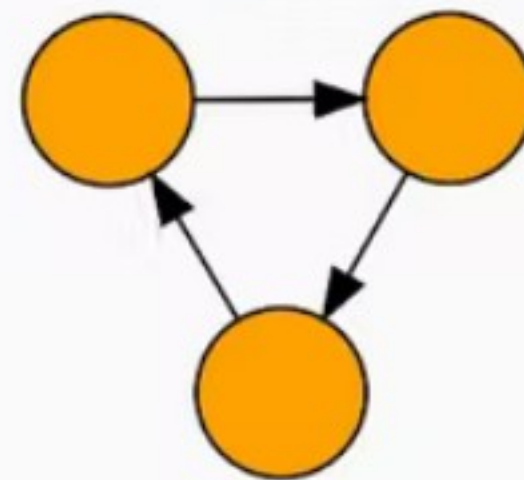
Veier og Stier

# Sykler

- *Sykel*: sti i en graf med minst tre noder, som forbinder første og siste node



- Grafer uten sykler kalles *asykliske*.
- i rettede grafer: kantene må “peke i riktig retning”



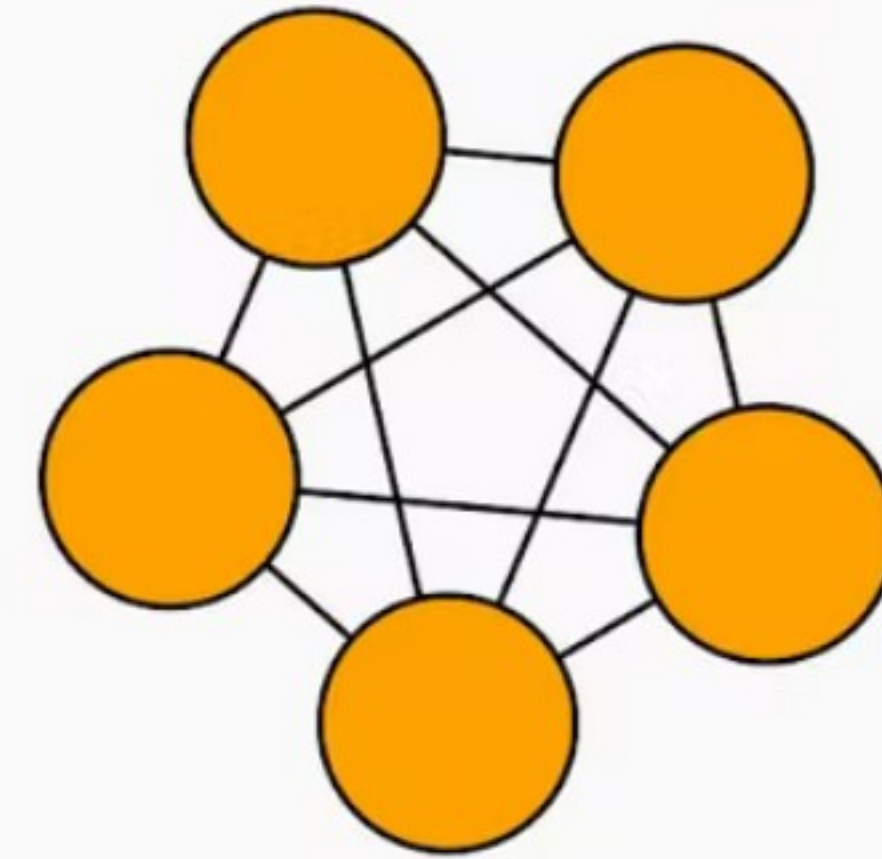
Sykler





## Estimere størrelsen av grafer

- En *komplett* graf er en graph med en kant mellom hvert par av noder
- Det blir tilsammen  $\frac{|V| \cdot (|V| - 1)}{2}$  kanter (uten løkker)
- Dermed er  $|E|$  i  $O(|V|^2)$
- Antall noder er altså et nyttig estimat for størrelsen av en graf
- når vi analyserer algoritmer, gjør vi det avhengig av både  $|E|$  og  $|V|$ , men verdt å huske at antall kanter er begrenset av antall noder (men ikke motsatt!)



Estimere størrelsen av grafer







# Viktige outcomes:

Vektet vs Uvektet kanter

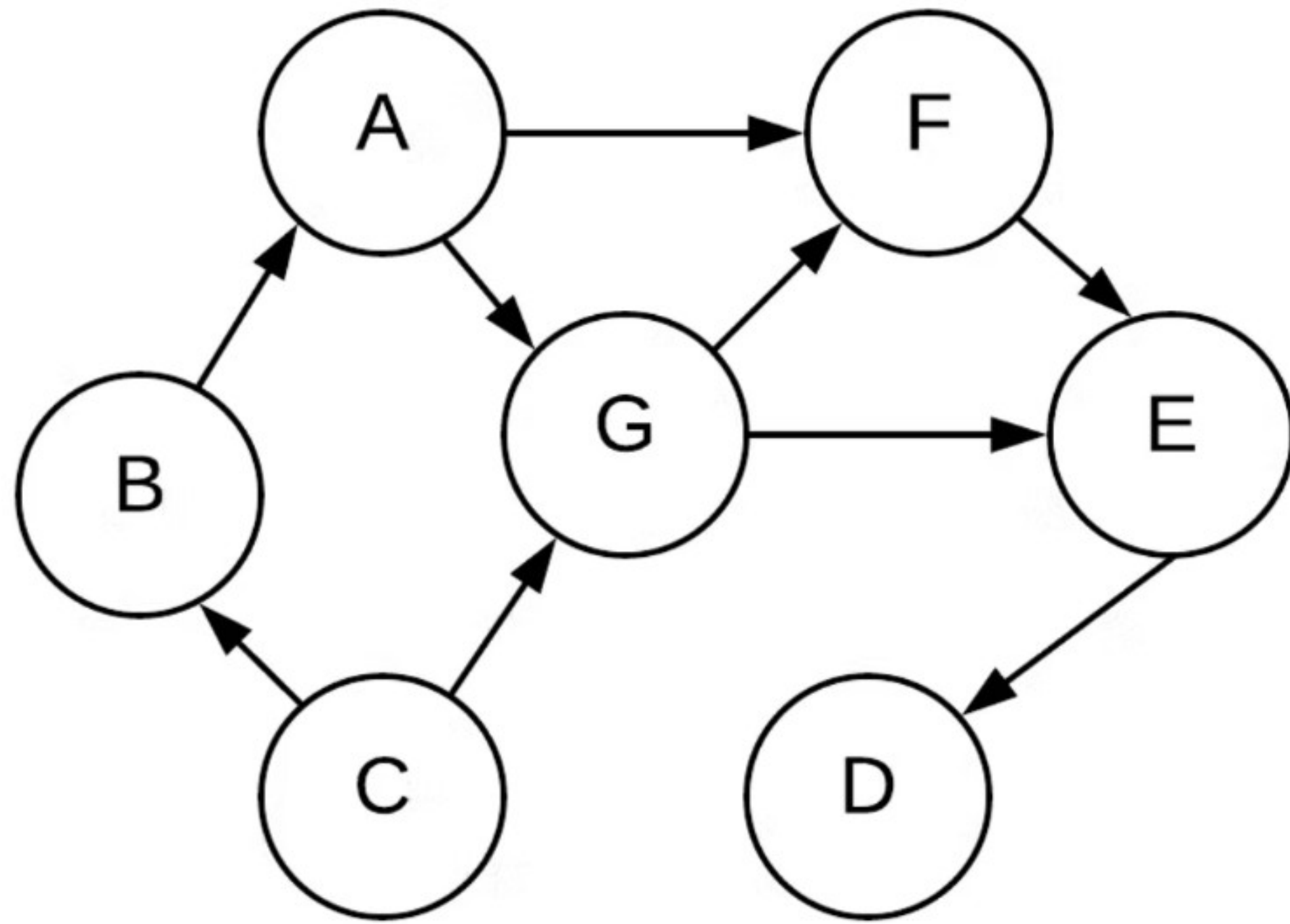
Rettete vs Urettede grafer, og hvordan de representeres

Vei vs Sti

Sykler og Asykliske grafer

Komplette grafer

Hvordan vi kan representere grafer.



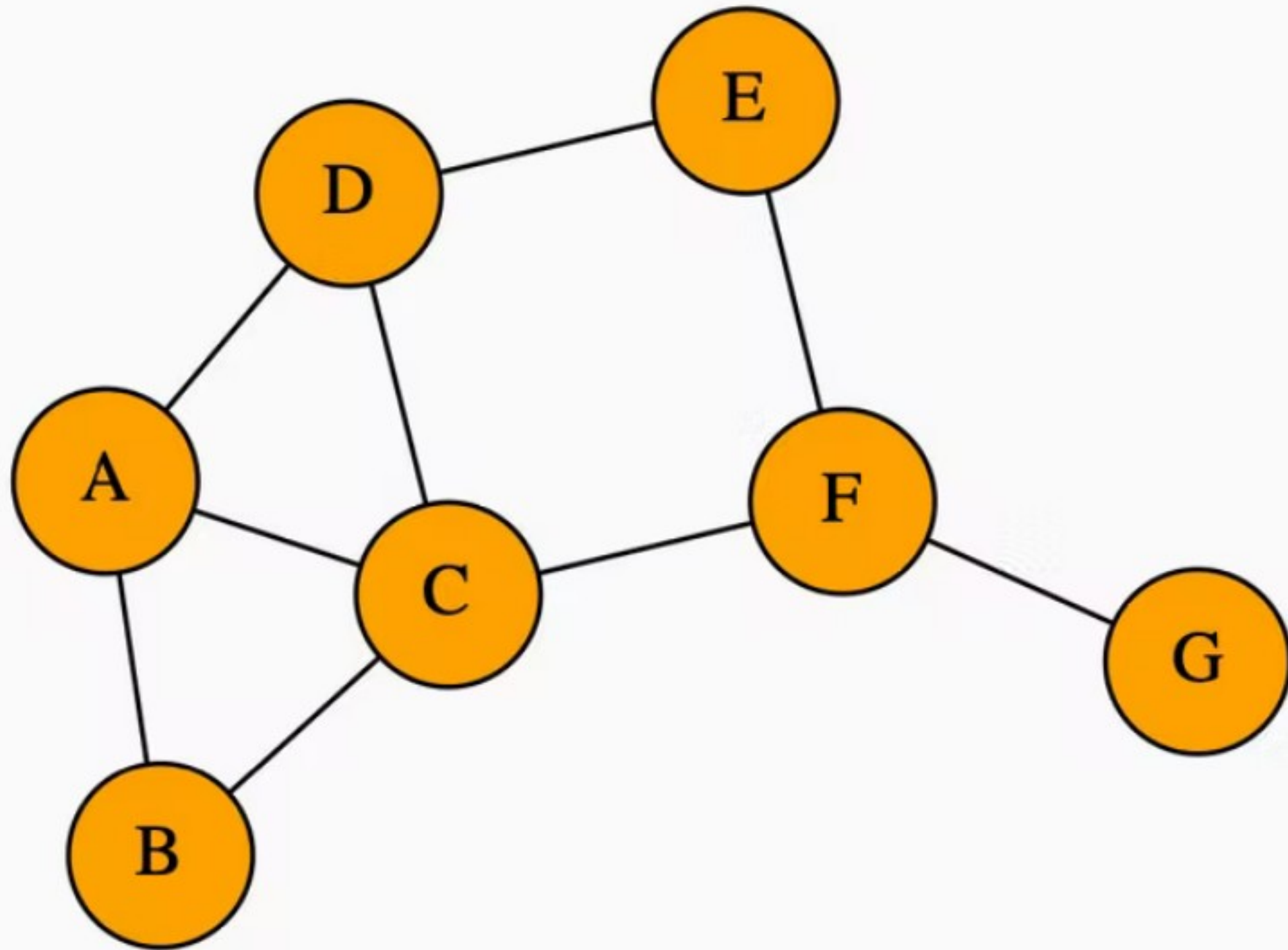
# Grafteraversering

DFS og BFS

Viktig note! Traversering handler kun om å gå gjennom e grav.  
Aka å komme fra et punkt i grafen til et annet  
Eller se hvor mange/hvilke noder som kan nås fra en gitt node.



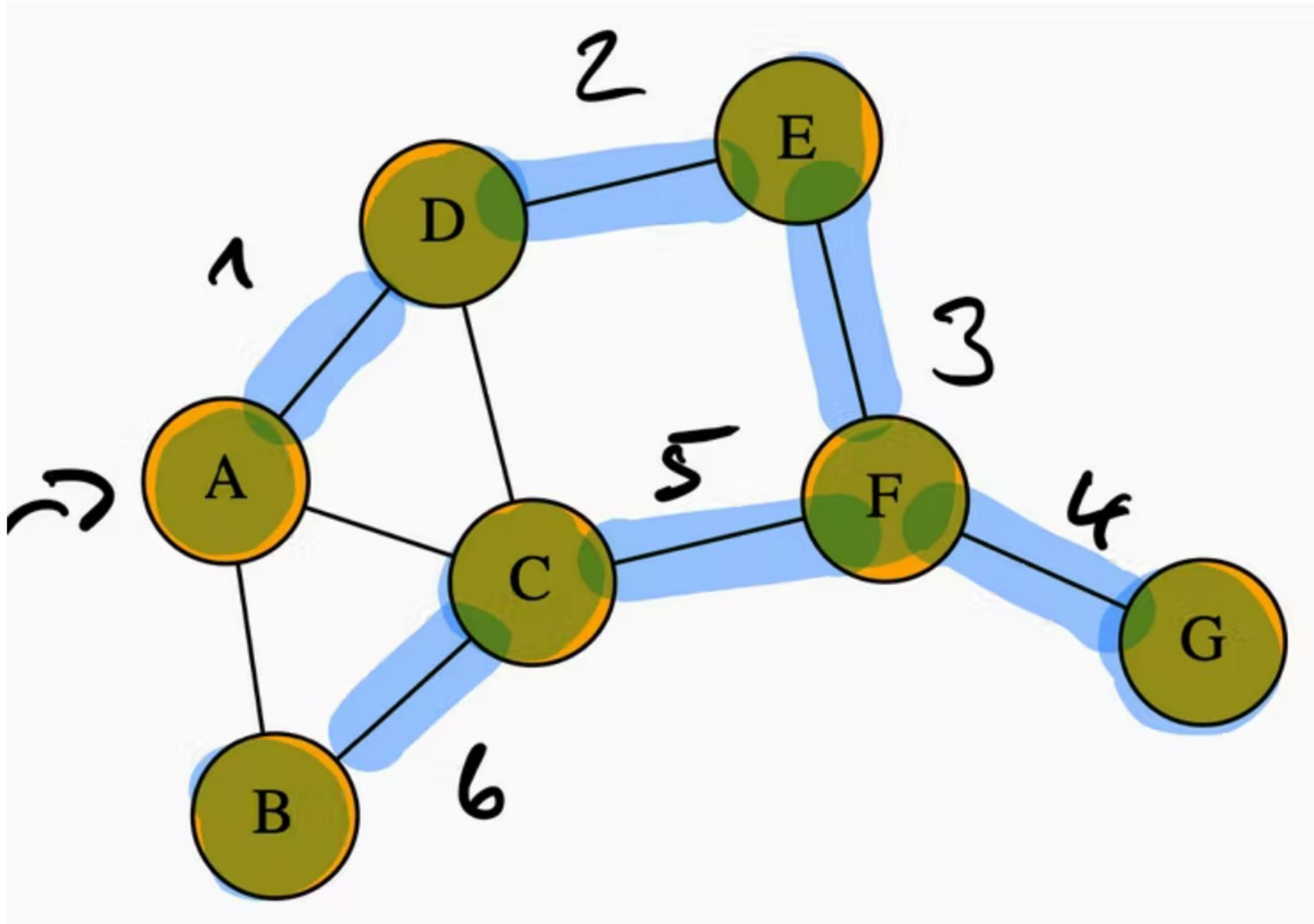




## Dybde først søk

- Ide: Gå så dypt som mulig
- Søker grafen nodevis
- For hver node, gå til neste nabonode
- Gjør dette så langt du kan
- Backtrack til tidligste node med en annen nabo



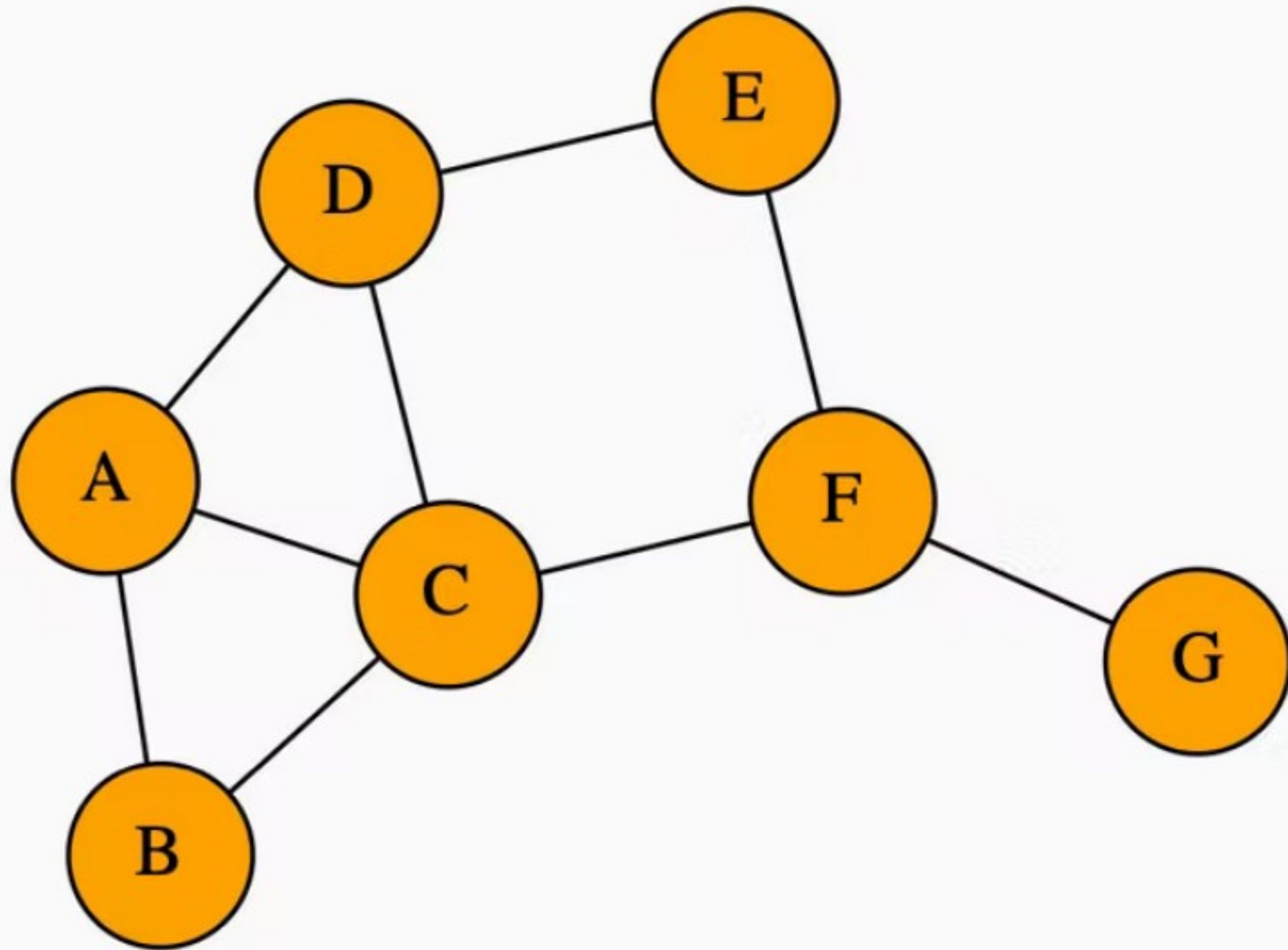


DFS



# DFS Visualisering

<https://visualgo.net/en/dfsdfs>

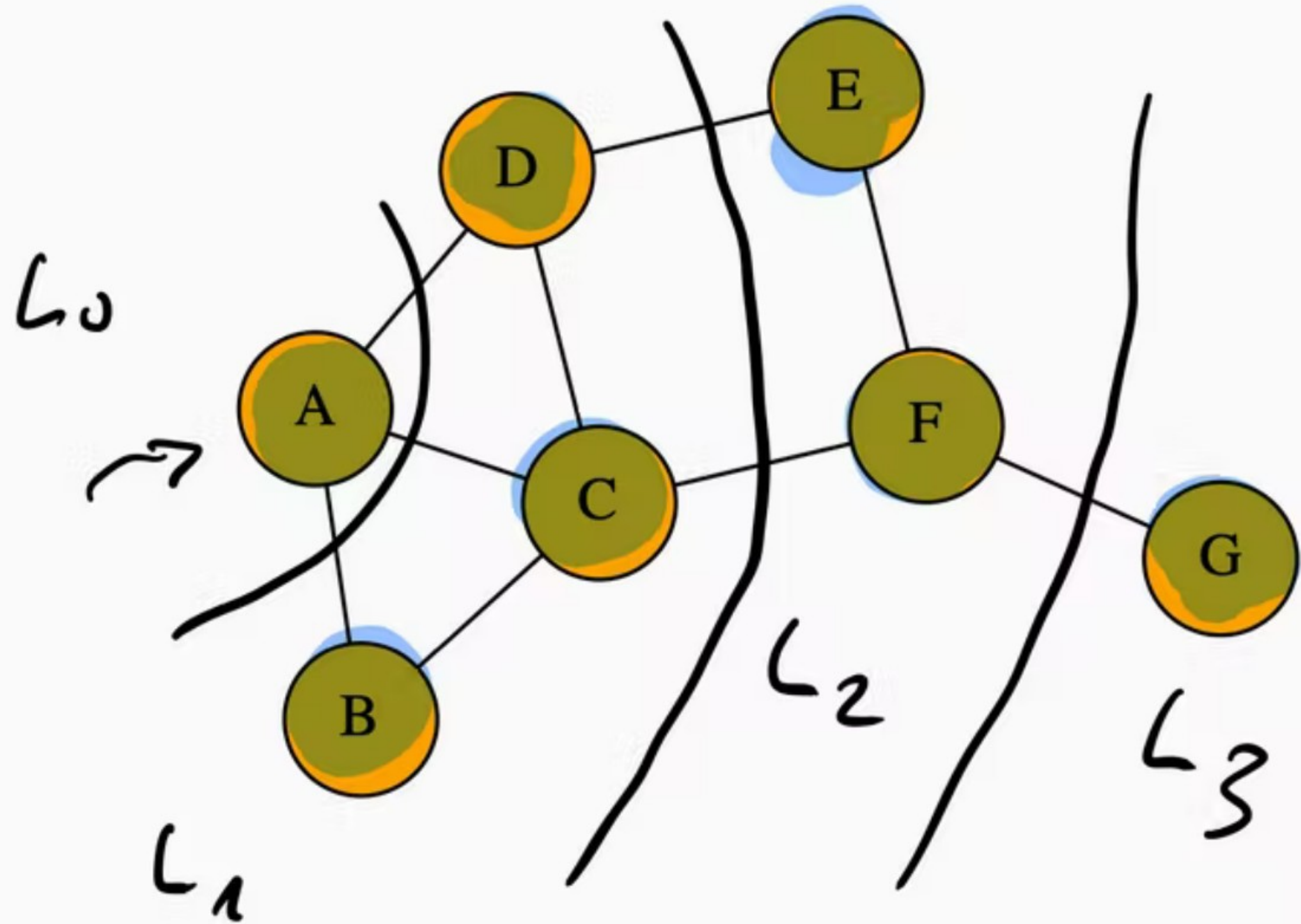


# Bredde først søk

- Ide: Dele grafen i ulike lag, og søker nodene i hvert lag
- Søker grafen lagvis







BFS

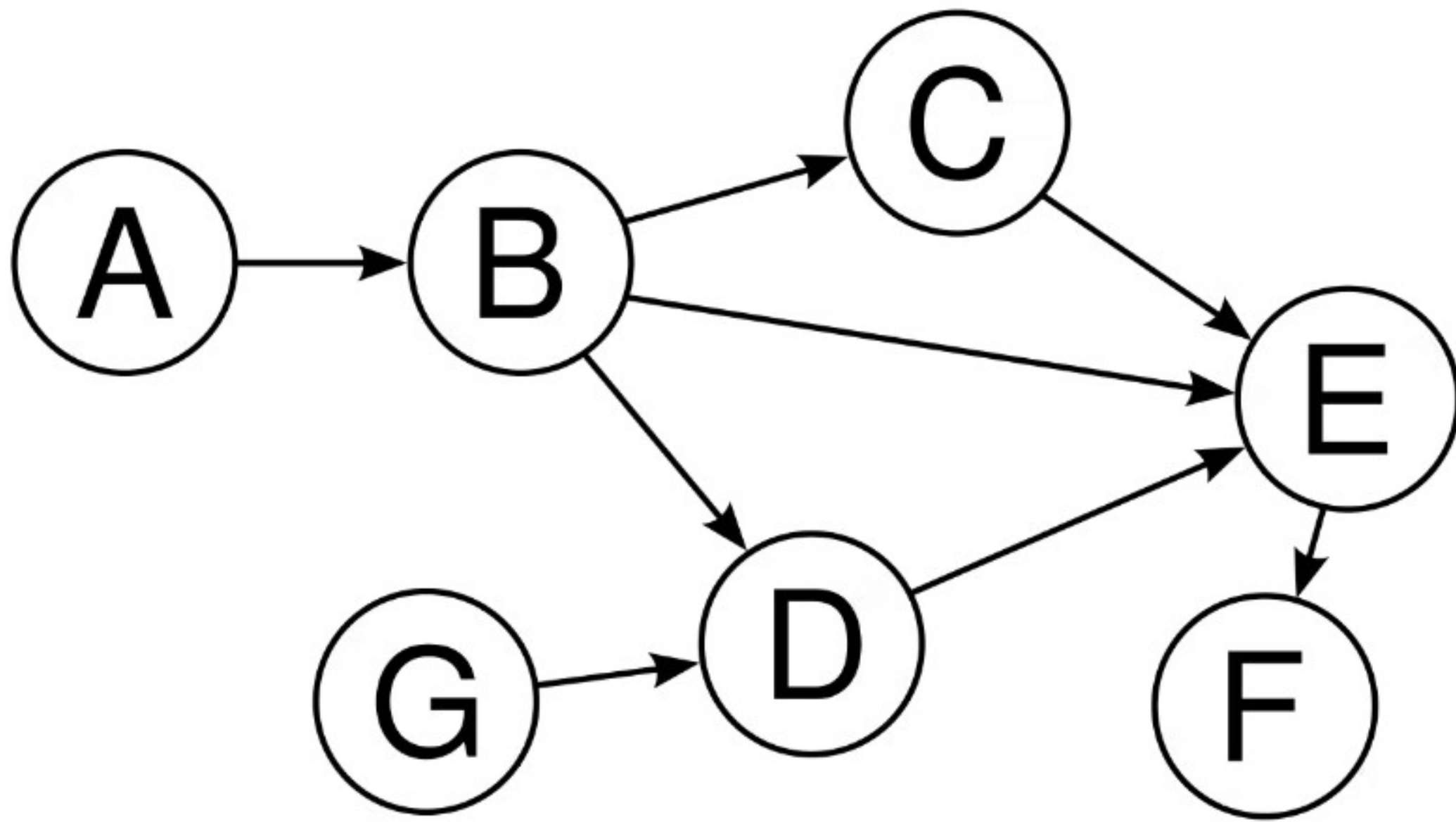




# BFS visualisering

<https://visualgo.net/en/dfsdfs>





# Topologisk Sortering

- Gleder for Rette Asykliske Grafer(DAG)
- Startnode må ha Inngrad
- Ide: Ta utgangspunkt i noder som ikke har noen avhengigheter/inngard 0

# Topologisk Sortering visualisering

<https://www.youtube.com/watch?v=clBFEhD77b4>



# Live koding

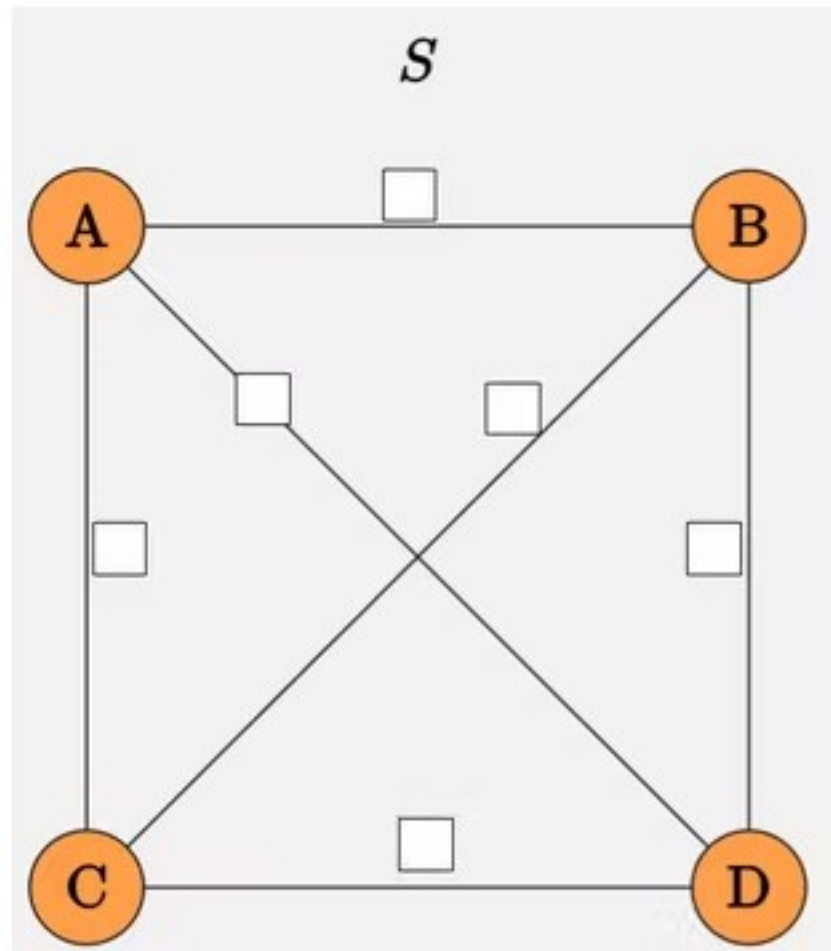




# Gruppeoppgaver

Anbefales!





# 4 Typer grafer

<https://www.uio.no/studier/emner/matnat/ifi/IN2010/h21/eksamensressurser/in2010-h2020-eksamen.pdf>

Side 14 og nedover



## Whops!-logger

10 poeng

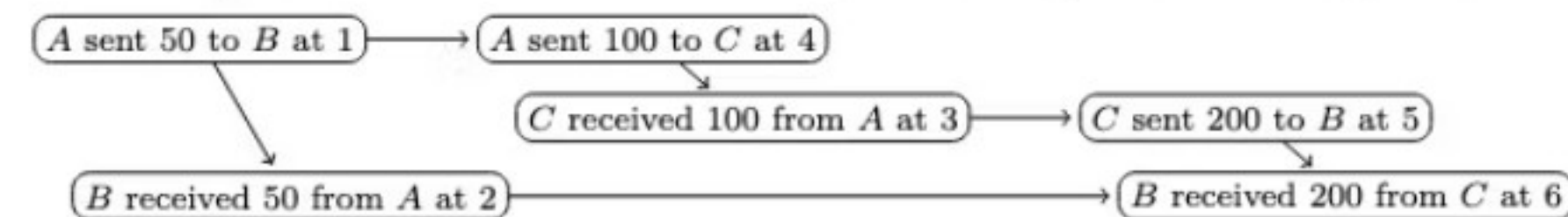
Tjenesten Whops! lar brukere overføre penger mellom hverandre gjennom en app. De prøver i så stor grad som mulig å unngå å lagre informasjon om brukerne sine sentralt, og lar heller informasjonen lagres på brukeren sin egen mobiltelefon.

En gang i blant har de behov for å anskaffe en komplett logg av overføringer for en periode for å kunne feilsøke systemene sine. Da henter de inn *lokale* logger fra noen utvalgte mobiltelefoner i systemet. En lokal logg er en liste av hendelser, der hver hendelse beskriver enten at et beløp er sent eller mottatt. I tillegg inneholder hver hendelse et tidsstempel som forteller når beløpet ble sendt eller mottatt. For enkelhets skyld er tidsstemplene representert som positive heltall. Her er et lite eksempel på tre logger fra mobiltelefoner  $A$ ,  $B$  og  $C$ .

$A$	$B$	$C$
$A$ sent 50 to $B$ at 1	$B$ received 50 from $A$ at 2	$C$ received 100 from $A$ at 3
$A$ sent 100 to $C$ at 4	$B$ received 200 from $C$ at 6	$C$ sent 200 to $B$ at 5

Dessverre er tidsstemplene fra mobiltelefonene upålitelige. De observerer ofte at et mottak av et beløp forekommer *før* beløpet ble sendt, i følge tidsstemplene. I eksempelet ovenfor mottar  $C$  et beløp på 100 *før*  $A$  har sendt beløpet til  $C$ .

For å approksimere en riktig logg lar de hver hendelse fra de lokale loggene være noder i en rettet graf  $G = (V, E)$ . Dersom to hendelser  $u$  og  $v$  kommer direkte etter hverandre i en lokal logg, så finnes det en rettet kant  $(u, v)$  i grafen. For hver hendelse  $u$  som representerer at et beløp er sendt, så finnes det en rettet kant  $(u, v)$  der  $v$  er hendelsen for det korresponderende mottaket av det beløpet. Her er grafen for eksempelet ovenfor:



Den ønskede komplette loggen for dette eksempelet skal se slik ut:

```

A sent 50 to B at 1
B received 50 from A at 2
A sent 100 to C at 4
C received 100 from A at 3
C sent 200 to B at 5
B received 200 from C at 6
  
```

Hver node  $v$  kan aksessere tidsstempelen fra loggen ved  $v.ts$ . Skriv en algoritme som tar rettet graf  $G = (V, E)$  som input, og skriver ut alle noder i  $V$  i en rekkefølge slik at:

- Dersom det finnes en kant fra  $u$  til  $v$  så skal  $u$  skrives ut før  $v$ .
- $u$  skal skrives ut før  $v$  hvis  $u.ts < v.ts$ , så lenge det ikke bryter med det første kravet.

# Kattisoppgave: Eksamen 2021





# Oppgave 6(Eulerkrets)

En Eulerkrets i en graf  $G$  er en vei som starter og slutter i samme node og er innom alle kantene i  $G$  nøyaktig en gang. Lag en algoritme som finner ut om  $G$  inneholder en Eulerkrets. Output skal være true/false. Kjører algoritmen i polynomiell tid? Begrunn svaret. Hint: Det finnes en egenskap som har med graden til nodene i  $G$  å gjøre, som nøyaktig bestemmer om  $G$  har en Eulerkrets eller ikke.





# Oppgave 7(Hamiltonsykel)

En Hamiltonsykel i en en graf  $G$  er en sykel som inneholder hver node nøyaktig en gang. Lag en algoritme som løser Hamiltonsykel. Output skal være true/false. Hint: Enn så lenge har ingen klart å finne en algoritme som løser dette problemet i polynomiell tid, så ikke ta det så tungt om også din algoritme bruker eksponensiell tid.



# Oppgaver fra Boka

R-13.5

R-13.6

R-13.7

C-13.15

C-13.16

