Grafer



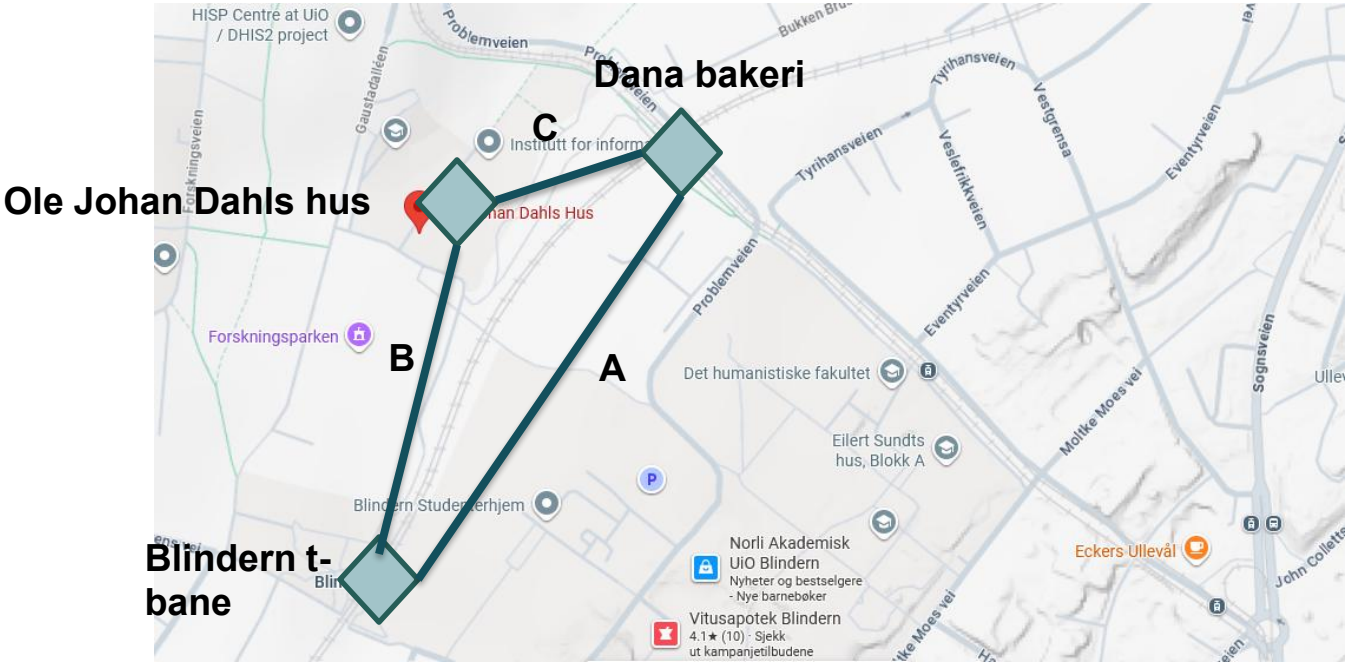
IN2010 – Uke 6

Hva er grafer?

En mengde med noder – V
En mengde med kanter – E

Hva er grafer?

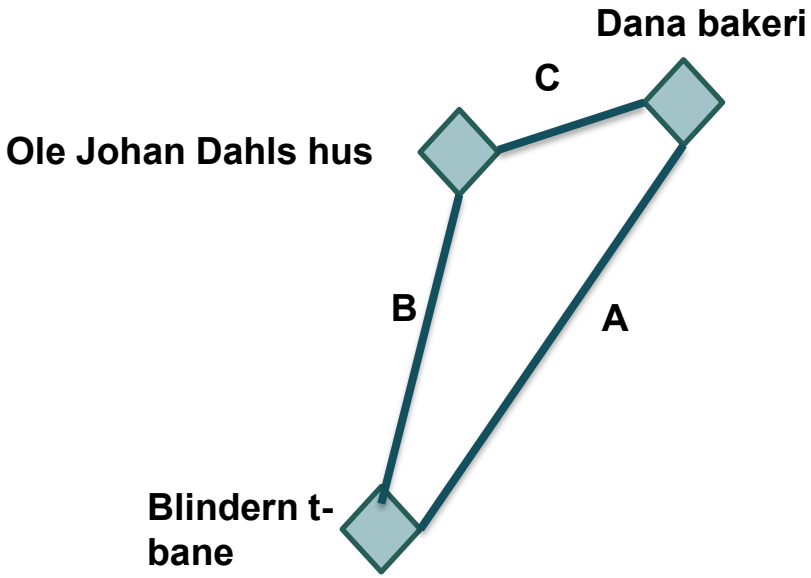
En mengde med noder – V
En mengde med kanter – E



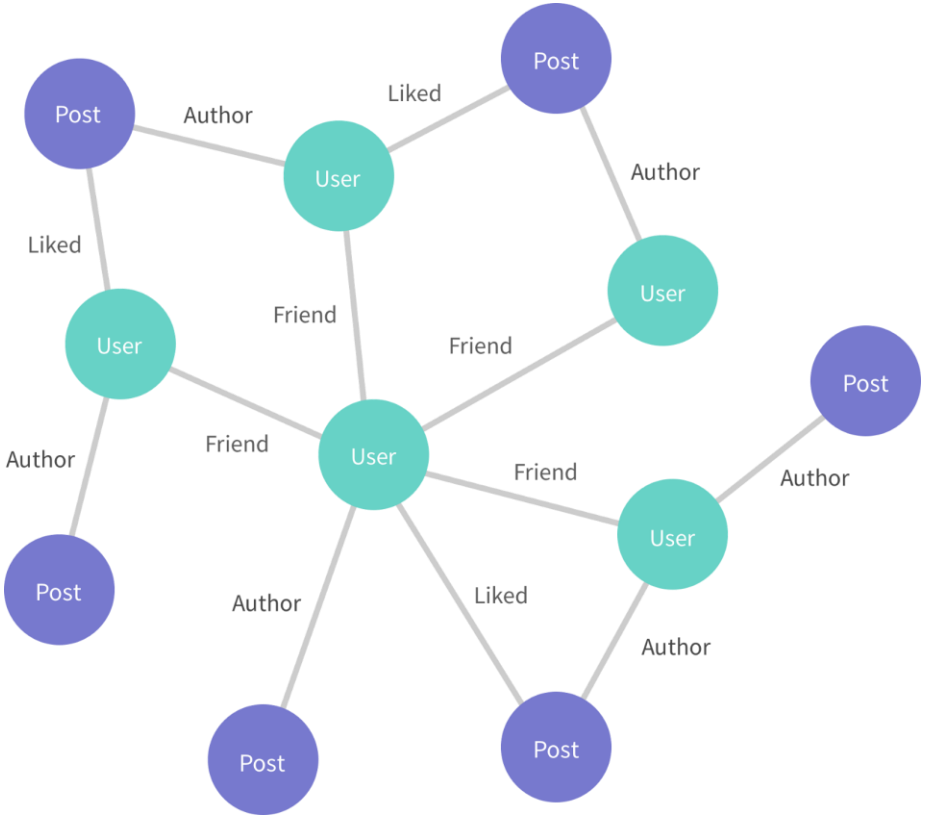
Hva er grafer?

En mengde med noder – V
En mengde med kanter – E

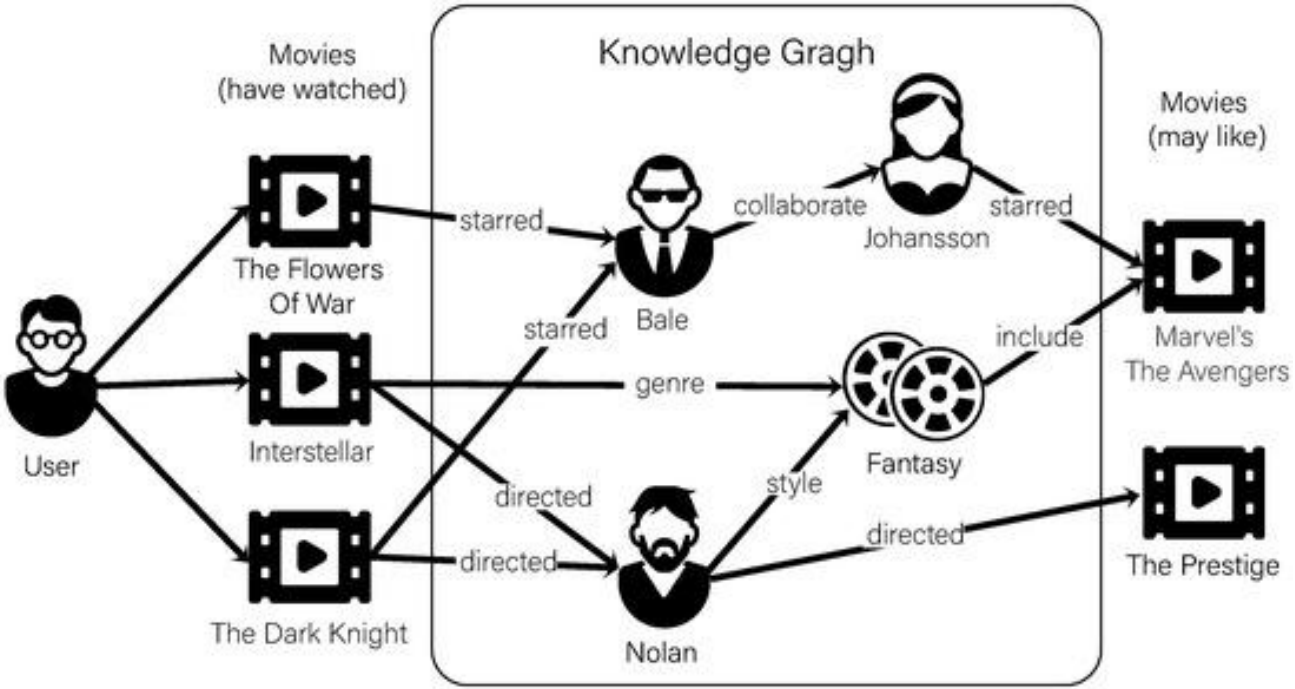
V = {Dana bakeri, OJD, Blindern T}
E = {A,B,C}



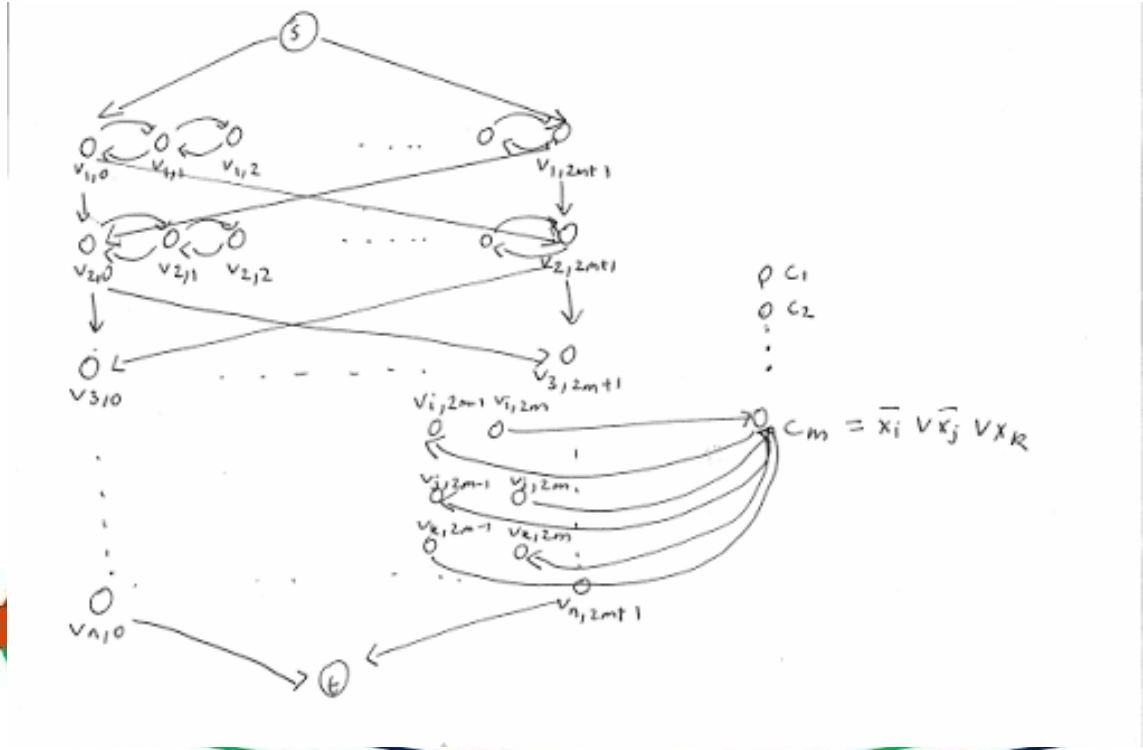
Hva er grafer?



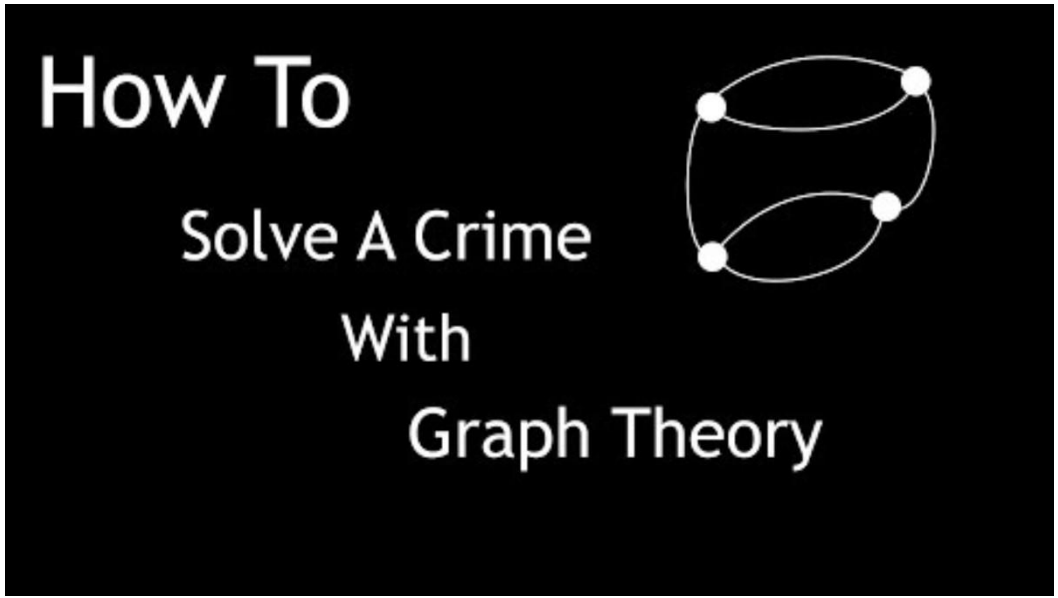
Hva er grafer?



Hva er grafer?



Ukas video!

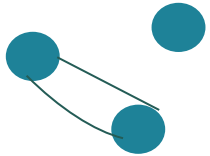


En mini-introduksjon til hvordan grafer kan brukes for å løse logiske problemer
(OBS: Personen prater litt fort)

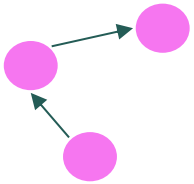
https://youtu.be/TwHy2DuWB3k

Memory refresh (2-3 min)

1) Vektet graf med en løkke

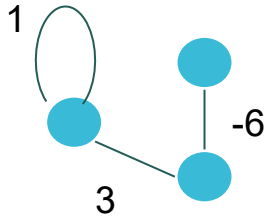


2) Rettet graf



3) Parallelle kanter

Hvilken av disse er sammenhengende?



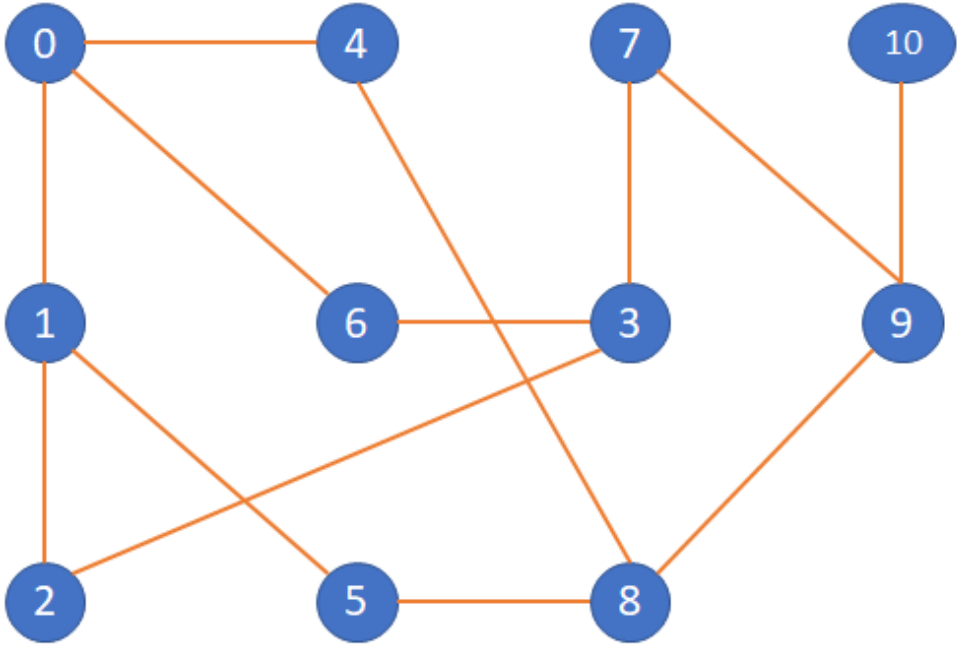
Veier vs stier

Vi bryr oss mest om stier!

Stier: Serie med noder der ingen **noder** (!) gjentas

Veier er det samme, men der ingen **kanter** kan gjentas

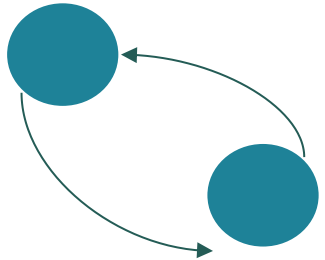
Hva er den korteste stien fra 0 til 10?



Sykler

En **sti** som har:
1) Minst tre noder
2) Der første og siste node i stien er ‘koblet sammen’

Altså: Kan du gå rundt i den!



OBS! I rettede grafer kan sykler bestå av kun to noder, men kantene må peke i riktig retning

Sykler

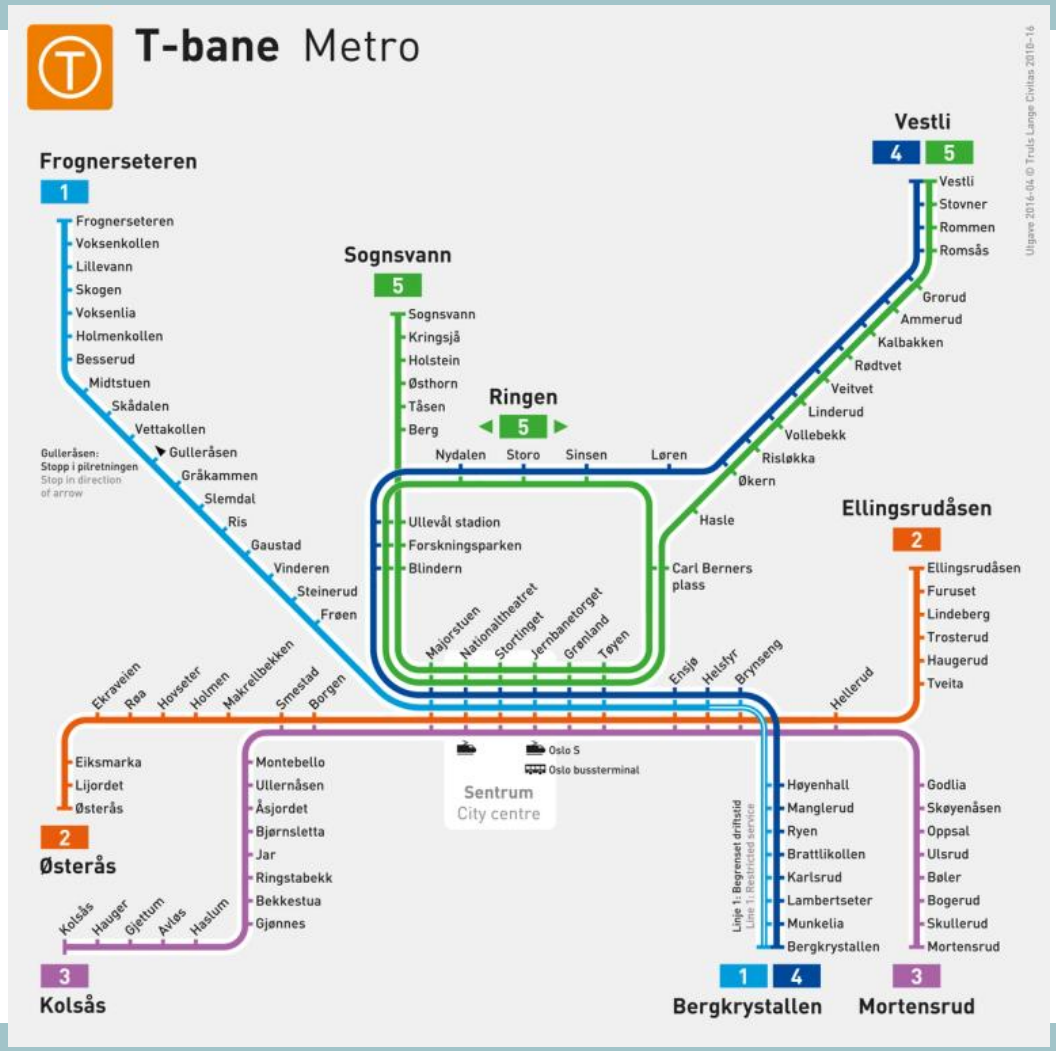
En **sti** som har:

1) Minst tre noder
2) Der første og siste node i stien er ‘koblet sammen’

Altså: Kan du gå rundt i den!



Sykler


T-bane Metro
Vestli
4 5
Vestli
Stovner
Rommen
Romsås
Grorud
Ammerud
Kalbakken
Rødtvet
Veitvet
Linderud
Vollebekk
Risløkka
Økern
Frognerseteren
1
Frognerseteren
Voksenkollen
Lillevann
Skogen
Voksenlia
Holmenkollen
Besserud
Midtstuen
Skådalen
Vettakollen
Gulleråsen
Gråkammen
Slemdal
Ris
Gaustad
Vinderen
Steinerud
Frøen
Gulleråsen: Stopp i pilretningen
Stop in direction of arrow
Sognsvann
5
Sognsvann
Kringsjå
Holstein
Østhorn
Tåsen
Berg
Ringen
5
Nydalen
Storo
Sinsen
Løren
Hasle
Ullevål stadion
Forskningsparken
Blindern
Carl Berners plass
Majorstuen
Nationaltheatret
Stortinget
Jernbanetorget
Grønland
Tøyen
Ensjø
Helsfyr
Brynseng
Hellerud
Ellingsrudåsen
2
Ellingsrudåsen
Furuset
Lindeberg
Trosterud
Haugerud
Tveita
Ekraveien
Røa
Høvseter
Holmen
Makrellbekken
Smestad
Borgen
Oslo S
Oslo bussterminal
Sentrum
City centre
Eiksmarka
Lijordet
Østerås
2
Østerås
Montebello
Ullernåsen
Åsjordet
Bjørnsletta
Jar
Ringstabekk
Bekkestua
Gjønnes
Kolsås
Hauger
Gjettum
Avløs
Haslum
3
Kolsås
Linje 1: Begrenset driftstid
Line 1: Restricted service
Høyenhall
Manglerud
Ryen
Brattlikollen
Karlsrud
Lambertseter
Munkelia
Bergkrystallen
1 4
Bergkrystallen
Godlia
Skøyenåsen
Oppsal
Ulsrud
Bøler
Bogerud
Skullerud
Mortensrud
3
Mortensrud
Utgave 2016-04 © Truls Lange Civitas 2010–16

Sykler

1) Er t-banen en sammenhengende graf?
2) Er det en komplett graf?
3) Hva er graden til noden ‘Brynseng’?



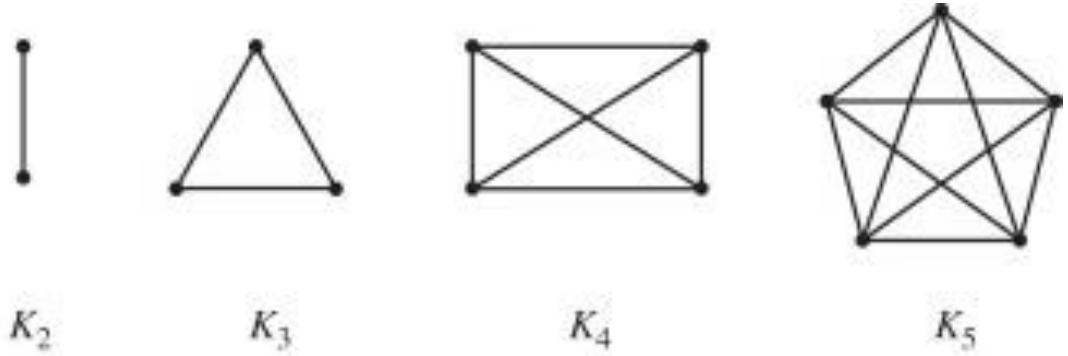
Littegrann skummel kompleksitet

- Når vi skriver algoritmer for grafer, varierer kjøretidskompleksiteten av både antall noder OG antall kanter
  - Dermed kan vi ikke bare skrive O(n), men heller O(|V| + |E|)

- Antall kanter er begrensa av antall noder, men vi kan som oftest ikke vite hvor mange kanter en graf her kun ved å vite antall noder
  - …Med mindre grafen er komplett

Littegrann skummel kompleksitet

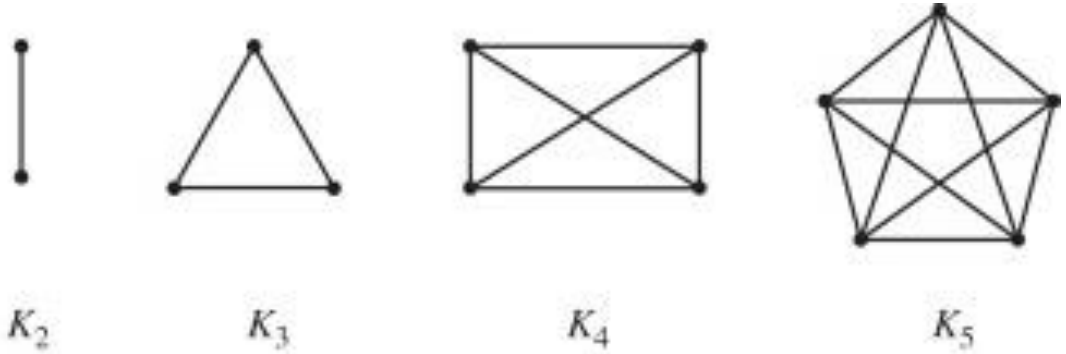
- Antall kanter er begrensa av antall noder, men vi kan som oftest ikke vite hvor mange kanter en graf her kun ved å vite antall noder
  - …Med mindre grafen er komplett



K_2 K_3 K_4 K_5

$$|E| = \frac{|V| * (|V| - 1)}{2}$$

Littegrann skummel kompleksitet



$$|E| = \frac{|V| * (|V| - 1)}{2}$$

Hvor mange kanter har en komplett graf med 6 noder?

Littegrann skummel kompleksitet

$$|E| = \frac{|V| * (|V| - 1)}{2} \longrightarrow |E| = O(|V|^2)$$

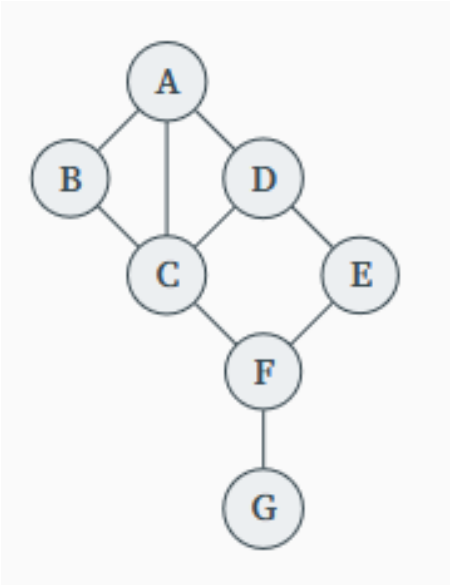
Et slags ‘worst case scenario’ (Størst case scenario)

Representasjon

«Ok, men hvordan programmerer vi det?»

Nabomatrise

	A	B	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	1	0	0	0	0
C	1	1	0	1	0	1	0
D	1	0	1	0	1	0	0
E	0	0	0	1	0	1	0
F	0	0	1	0	1	0	1
G	0	0	0	0	0	1	0



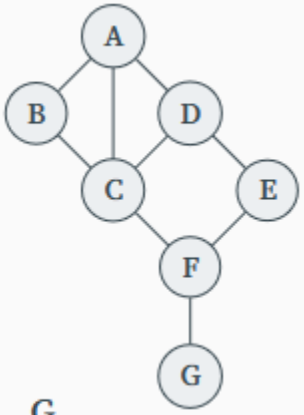
Naboliste

$A : [B, C, D]$
$B : [A, C]$
$C : [A, B, D, F]$
$D : [A, C, E]$
$E : [D, F]$
$F : [C, E, G]$
$G : [F]$

«Ok, men hvordan programmerer vi det?»

Nabomatrise

- 2D array
- Minne: $O(|V|^2)$
- Sjekke om to noder har en kant mellom seg? Konstant tid
- Finne alle naboene til en node? …vanskeligere

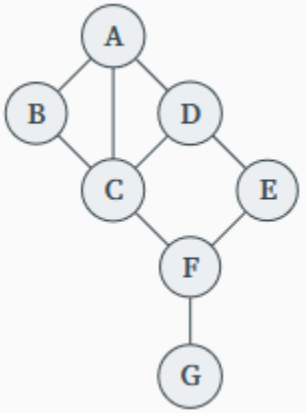


	A	B	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	1	0	0	0	0
C	1	1	0	1	0	1	0
D	1	0	1	0	1	0	0
E	0	0	0	1	0	1	0
F	0	0	1	0	1	0	1
G	0	0	0	0	0	1	0

«Ok, men hvordan programmerer vi det?»

Naboliste

- Dictionary
- Minne: O(|V| + |E|)
- Sjekke om to noder har en kant mellom seg? Kan gjøres i konstant tid – Hvordan?
- Finne alle naboene til en node? O(deg(v))



$$
\begin{aligned}
A &: [B, C, D] \\
B &: [A, C] \\
C &: [A, B, D, F] \\
D &: [A, C, E] \\
E &: [D, F] \\
F &: [C, E, G] \\
G &: [F]
\end{aligned}
$$

Graftraversering



Graftraversering

- Akkurat som med traversering av trær finnes det flere måter å ‘besøke’ alle noder
- ‘Grunnalgoritmer’

Dybde Først Søk (DFS)

Bredde Først Søk (DFS)

Dybde Først Søk (DFS)

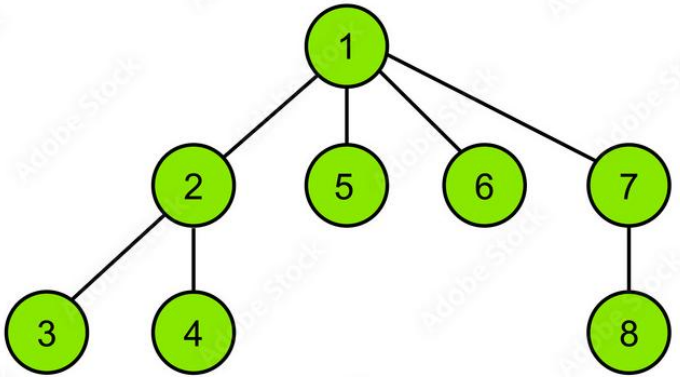
- Gå så dypt inn i grafen som mulig.
- Dette er sikkert sånn du hadde utforsket en labyrint
- Kan løses rekursivt eller iterativt (med en stack)

- DFSFull() og DFSVisit()
- O(|V| + |E|)

Dybde Først Søk (DFS)

Input: En graf $G = (V, E)$, en startnode s og en mengde `visited` med besøkte noder
Output: Besøker alle noder i G som kan nås fra s nøyaktig én gang

```
1 Procedure DFSVisit(G, s, visited)
2     stack ← stack containing s
3     while stack is not empty do
4         u ← stack.pop()
5         if u ∉ visited then
6             add u to visited
7             for (u, v) ∈ E do
8                 stack.push(v)
```



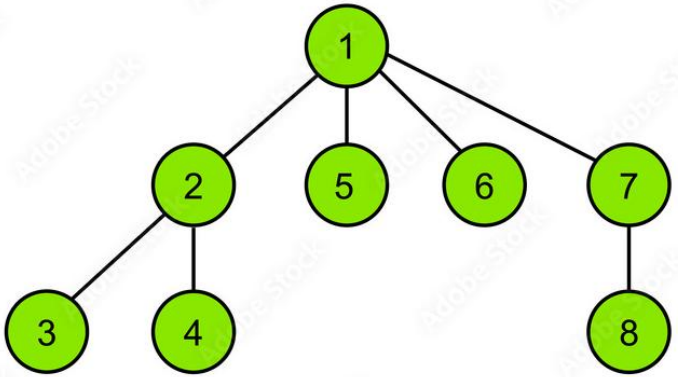
Bredde Først Søk (DFS)

- Ta de nærmest startnoden først
- Lagvis (som en løk), litt som når du kaster en stein i vannet

- Implementeres ofte med en kø
- O(|V| + |E|)

Bredde Først Søk (DFS)

```
Input: En graf G = (V, E) og en startnode s, og en
       mengde visited med besøkte noder
Output: Besøker alle noder i G som kan nås fra s
        nøyaktig én gang
1 Procedure BFSVisit(G, s, visited)
2     add s to visited
3     queue ← queue containing s
4     while queue is not empty do
5         u ← queue.dequeue()
6         for (u, v) ∈ E do
7             if v ∉ visited then
8                 add v to visited
9                 queue.enqueue(v)
```



Topologisk søk

DAG – Directed Acyclic Graph

- Rettet
- Ingen sykler

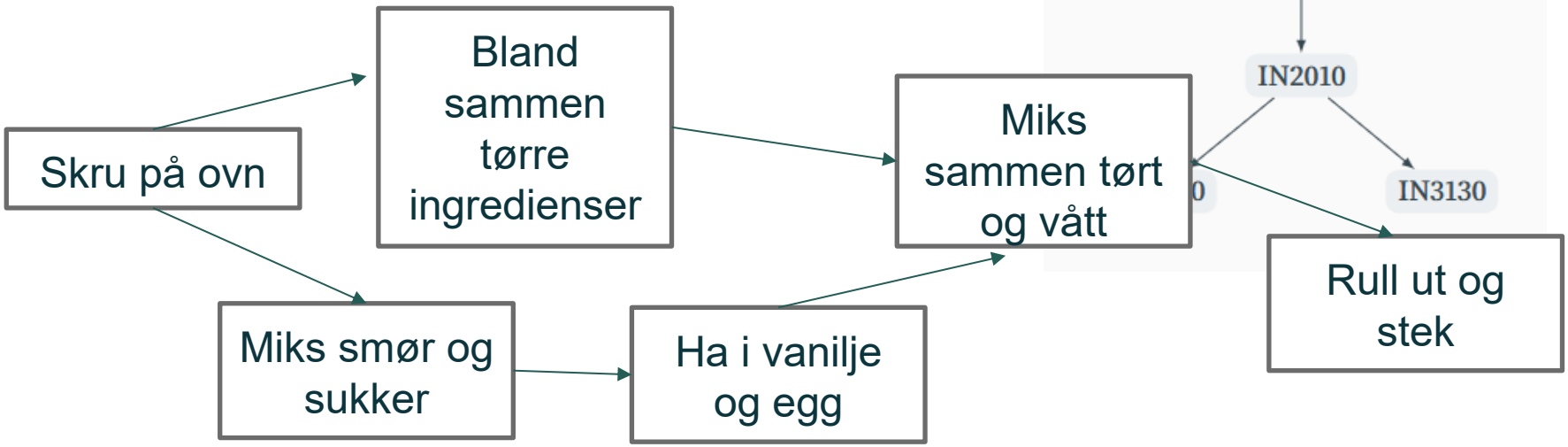
En slags sekvens med steg

Topologisk søk

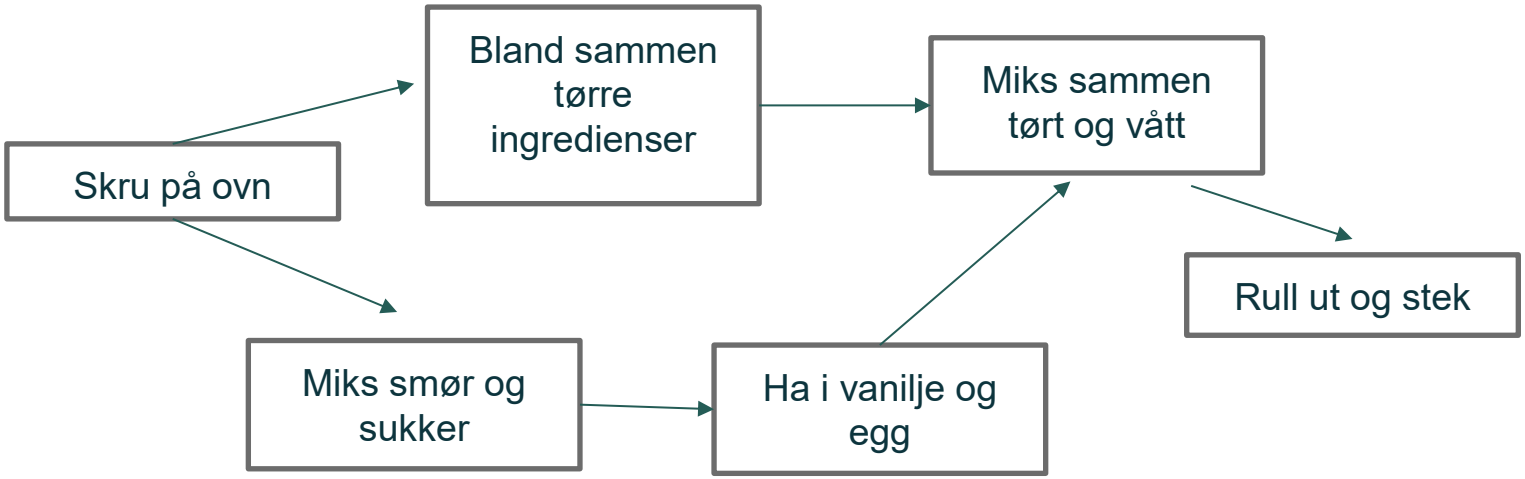
DAG – Directed Acyclic Graph

- Rettet
- Ingen sykler

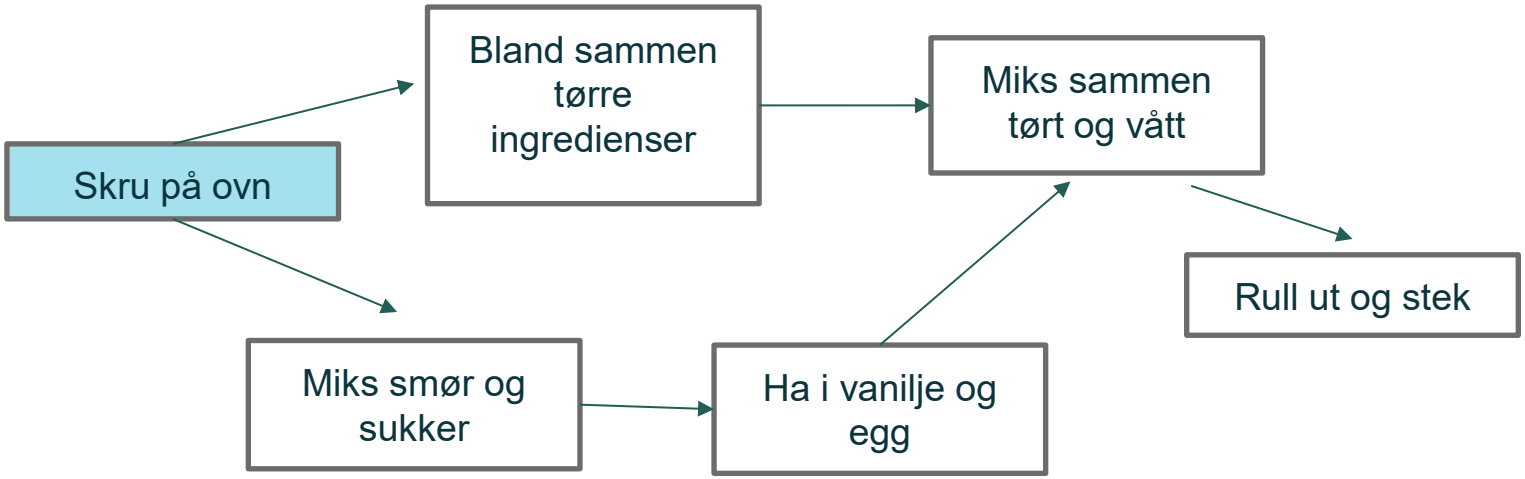
En slags sekvens med steg



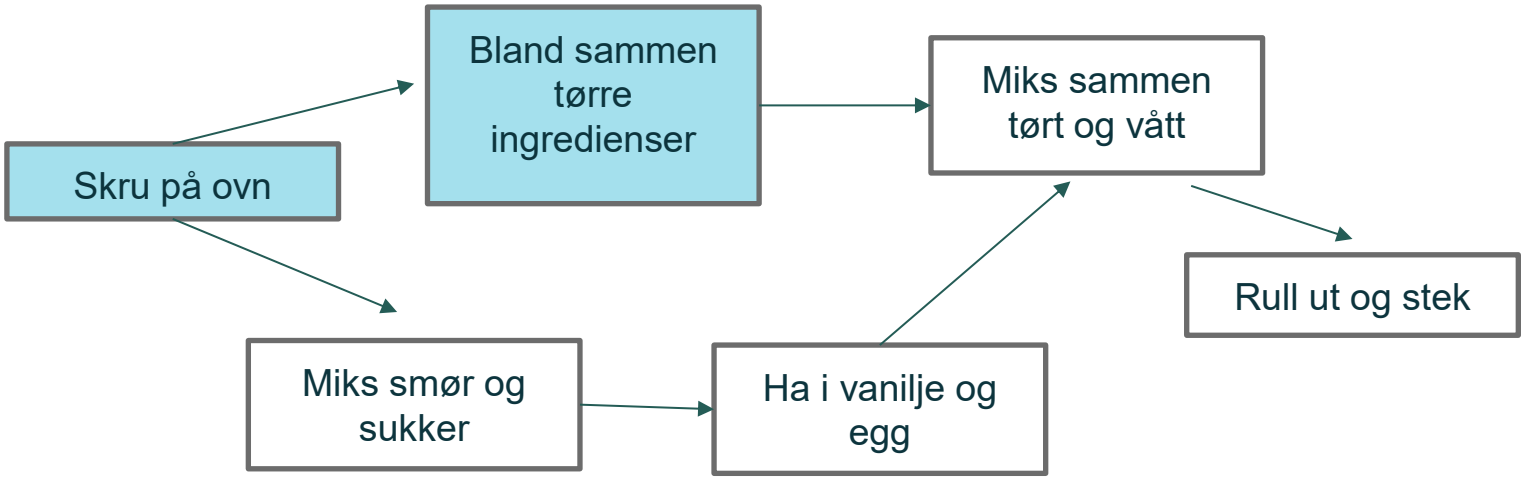
Topologisk sortering med DFS



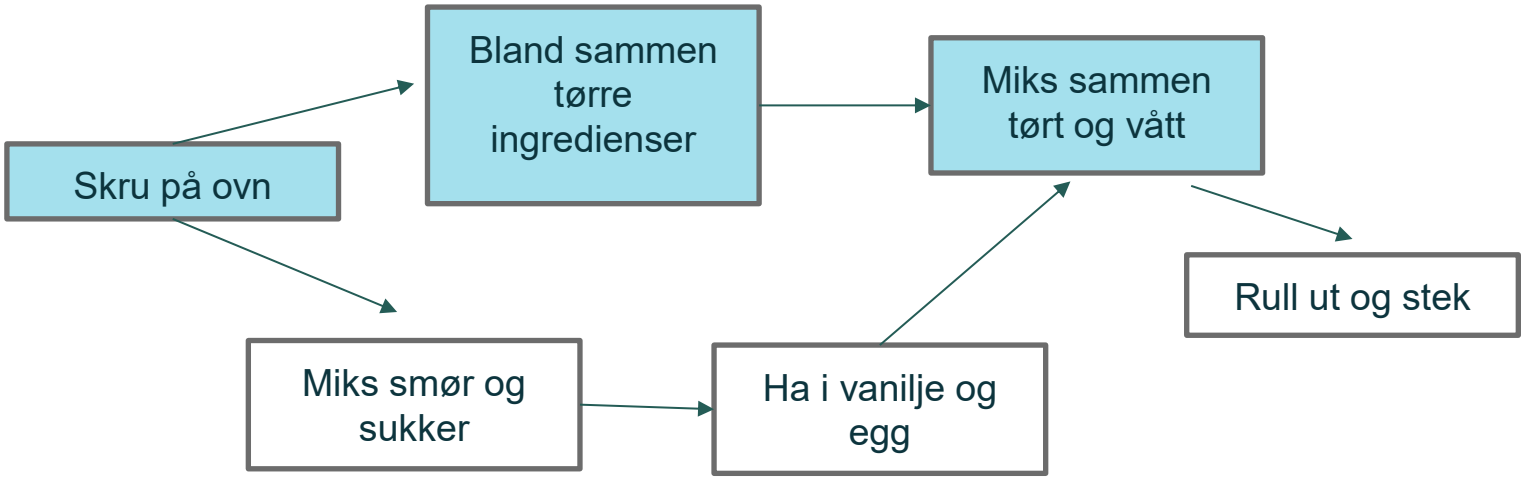
Topologisk sortering med DFS



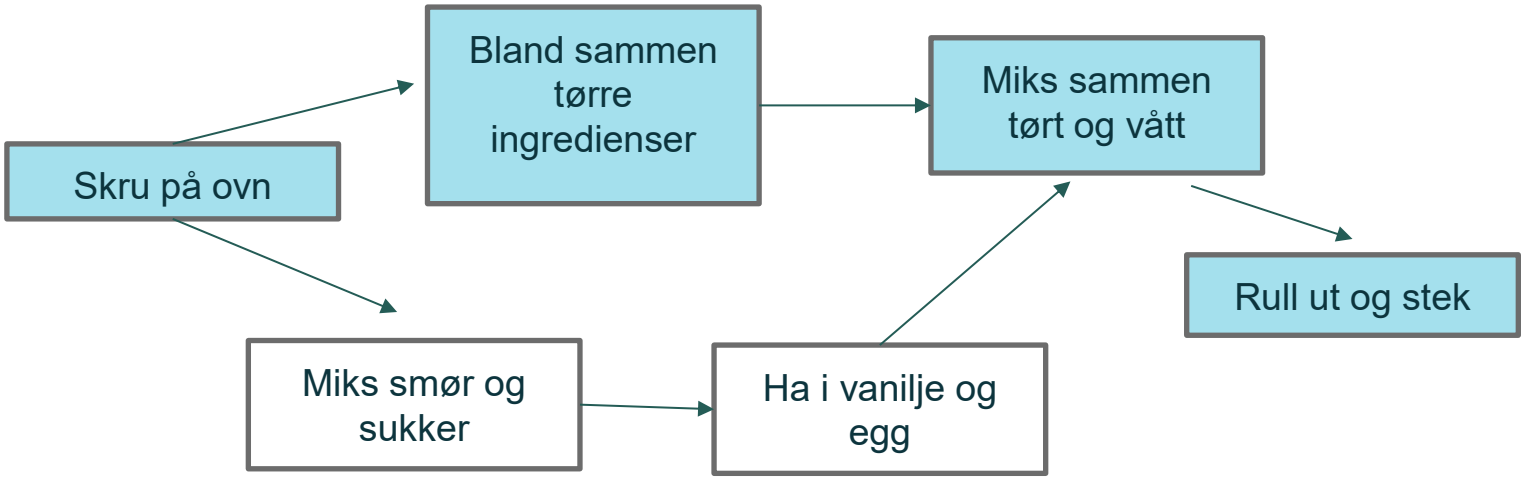
Topologisk sortering med DFS



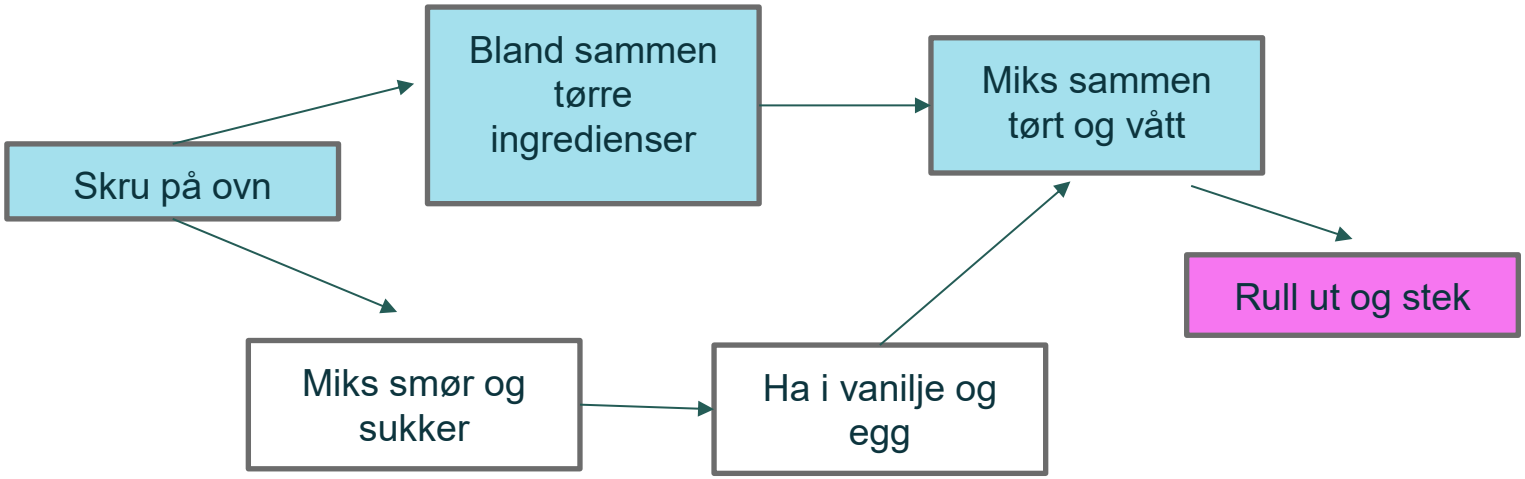
Topologisk sortering med DFS



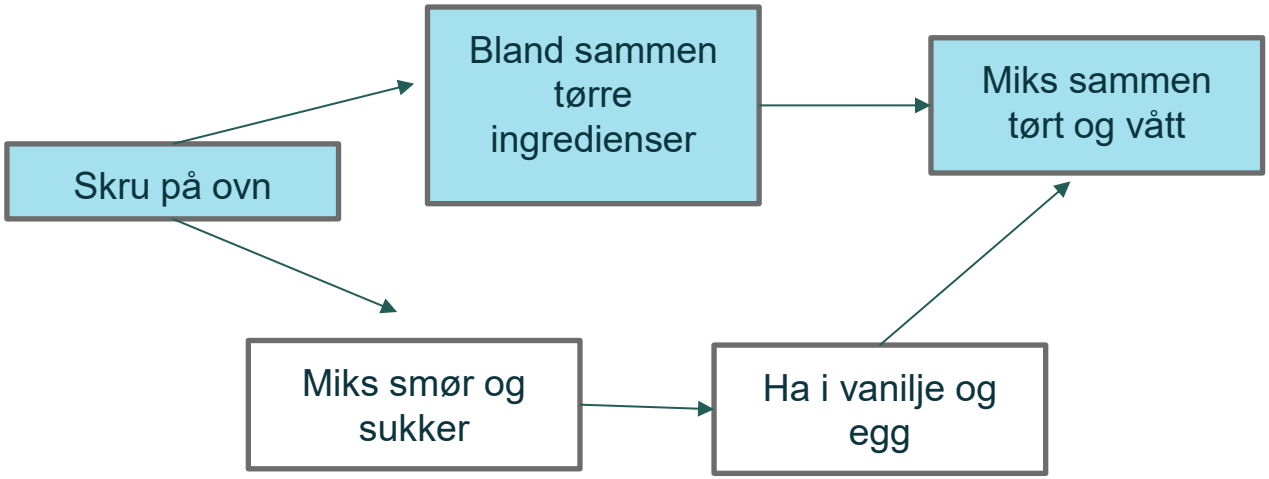
Topologisk sortering med DFS



Topologisk sortering med DFS

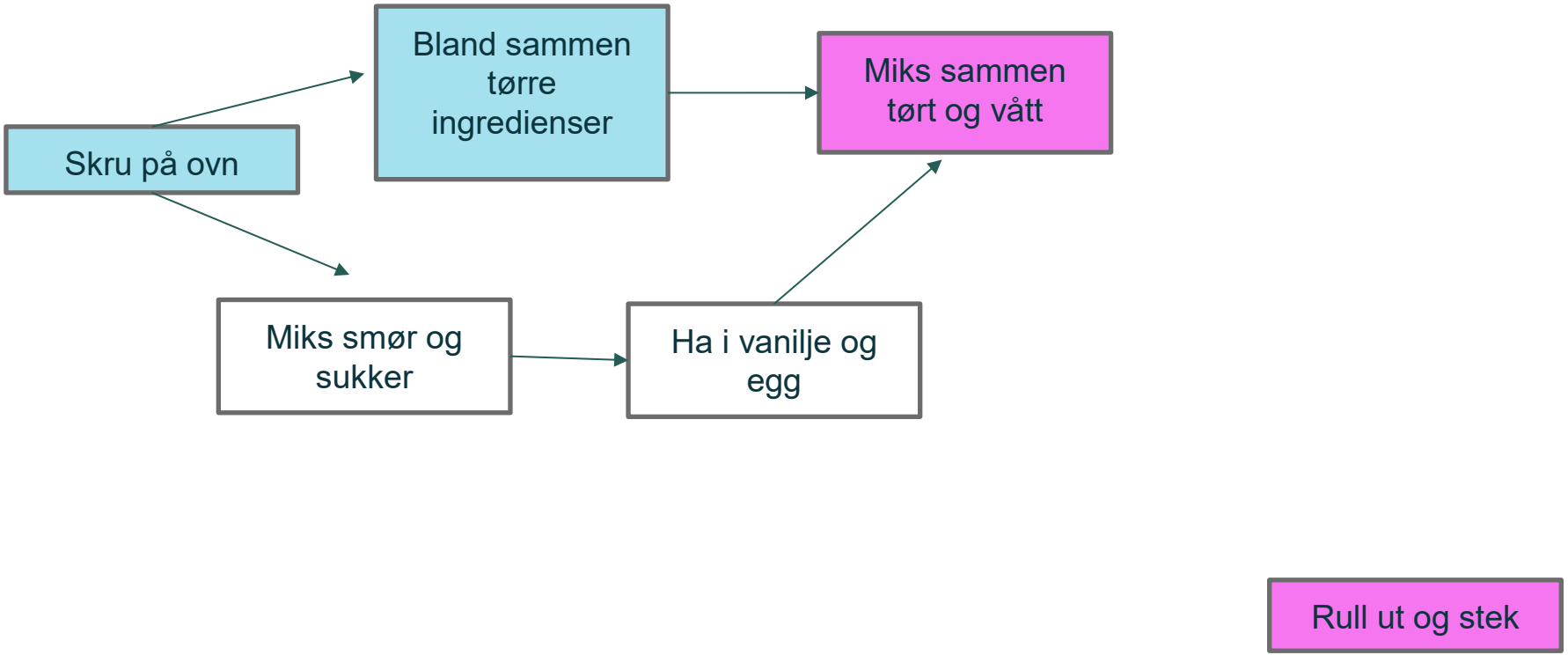


Topologisk sortering med DFS

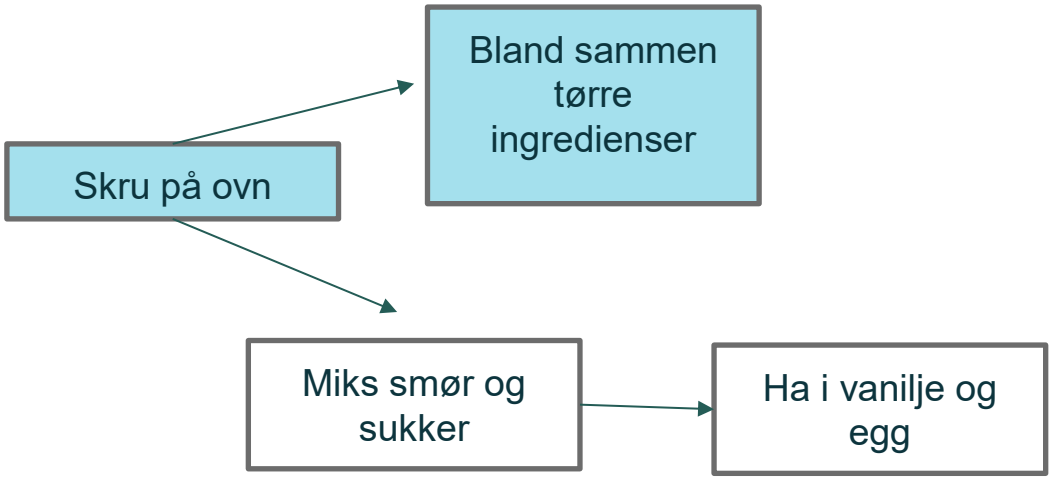


Rull ut og stek

Topologisk sortering med DFS



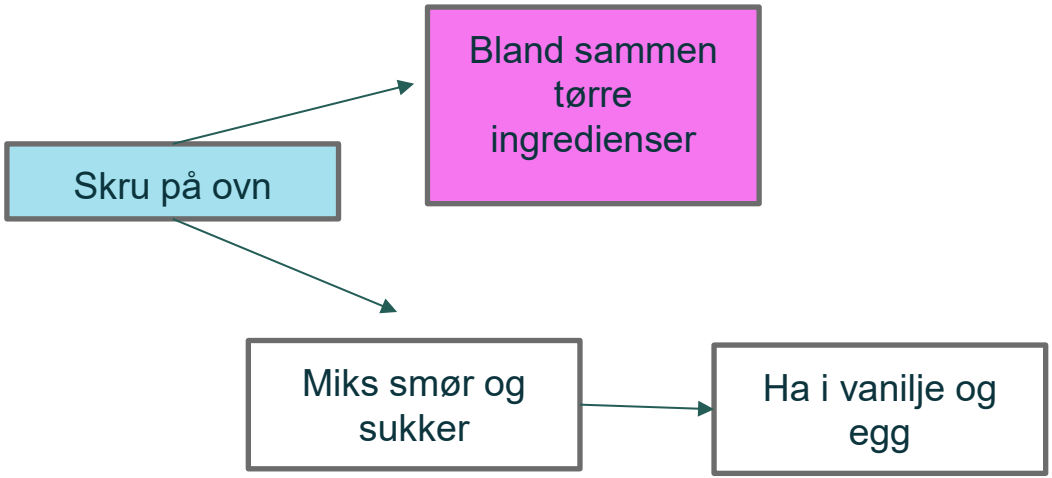
Topologisk sortering med DFS



Miks sammen tørt og vått

Rull ut og stek

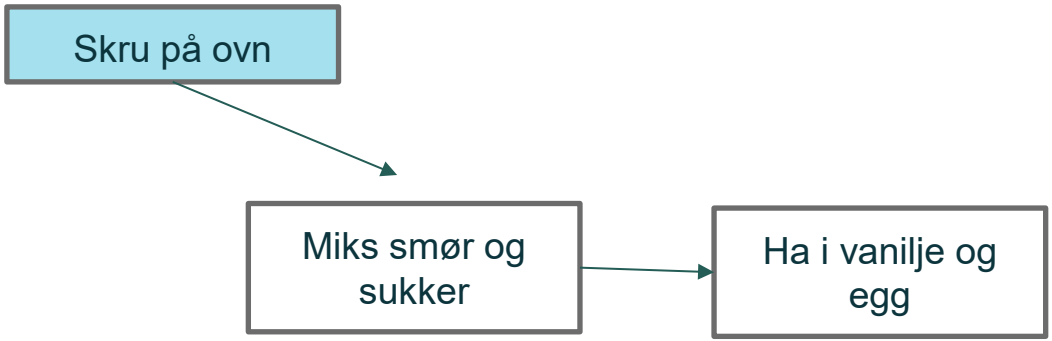
Topologisk sortering med DFS



Miks sammen tørt og vått

Rull ut og stek

Topologisk sortering med DFS

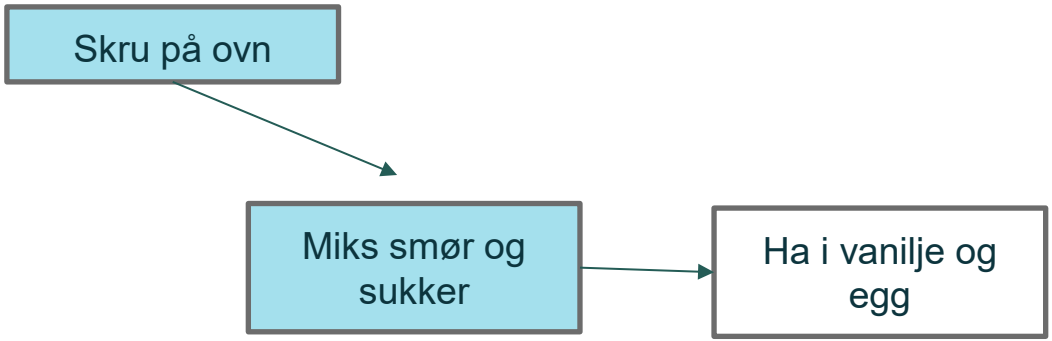


Bland sammen tørre ingredienser

Miks sammen tørt og vått

Rull ut og stek

Topologisk sortering med DFS

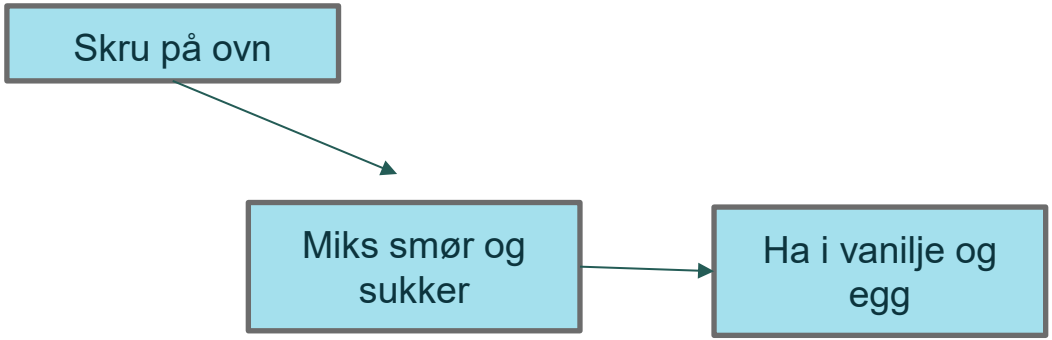


Bland sammen tørre ingredienser

Miks sammen tørt og vått

Rull ut og stek

Topologisk sortering med DFS

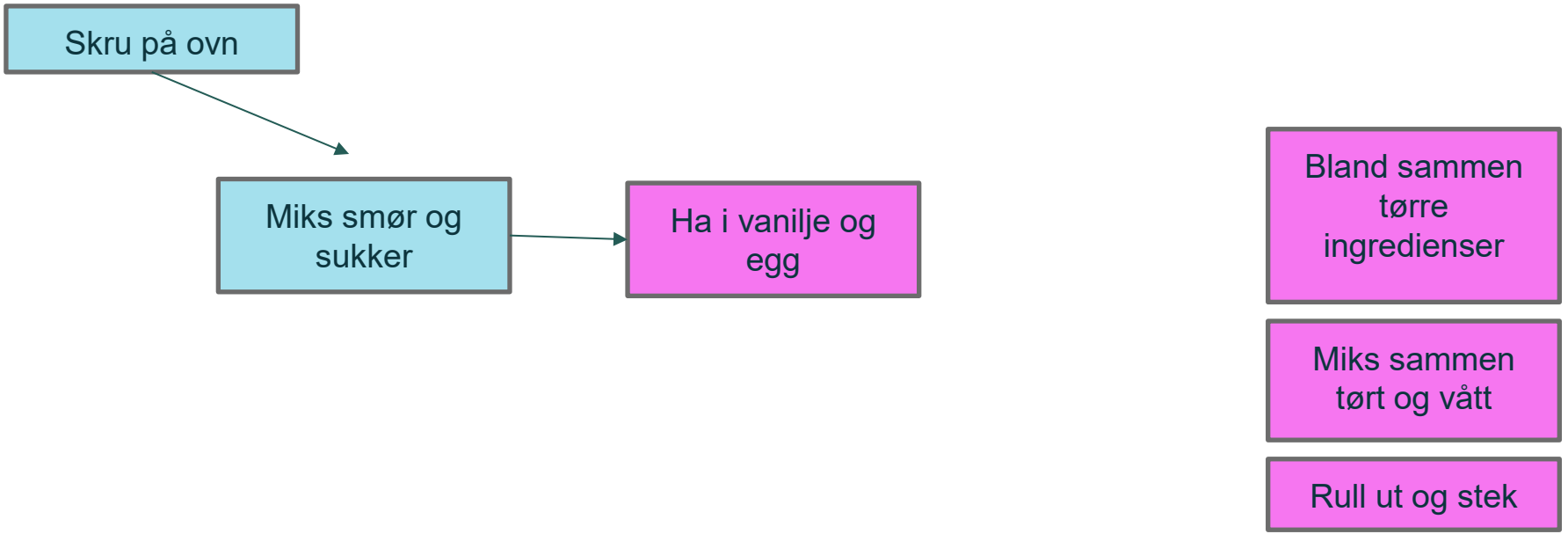


Bland sammen tørre ingredienser

Miks sammen tørt og vått

Rull ut og stek

Topologisk sortering med DFS



Topologisk sortering med DFS



Topologisk sortering med DFS



Topologisk sortering med DFS



Topologisk sortering med DFS



Topologisk sortering med DFS

1. Skru på ovn
2. Miks smør og sukker
3. Ha i vanilje og egg
4. Bland sammen tørre ingredienser
5. Miks sammen tørt og vått
6. Rull ut og stek

Oppgaver