Grafer

IN2010 gruppe 3

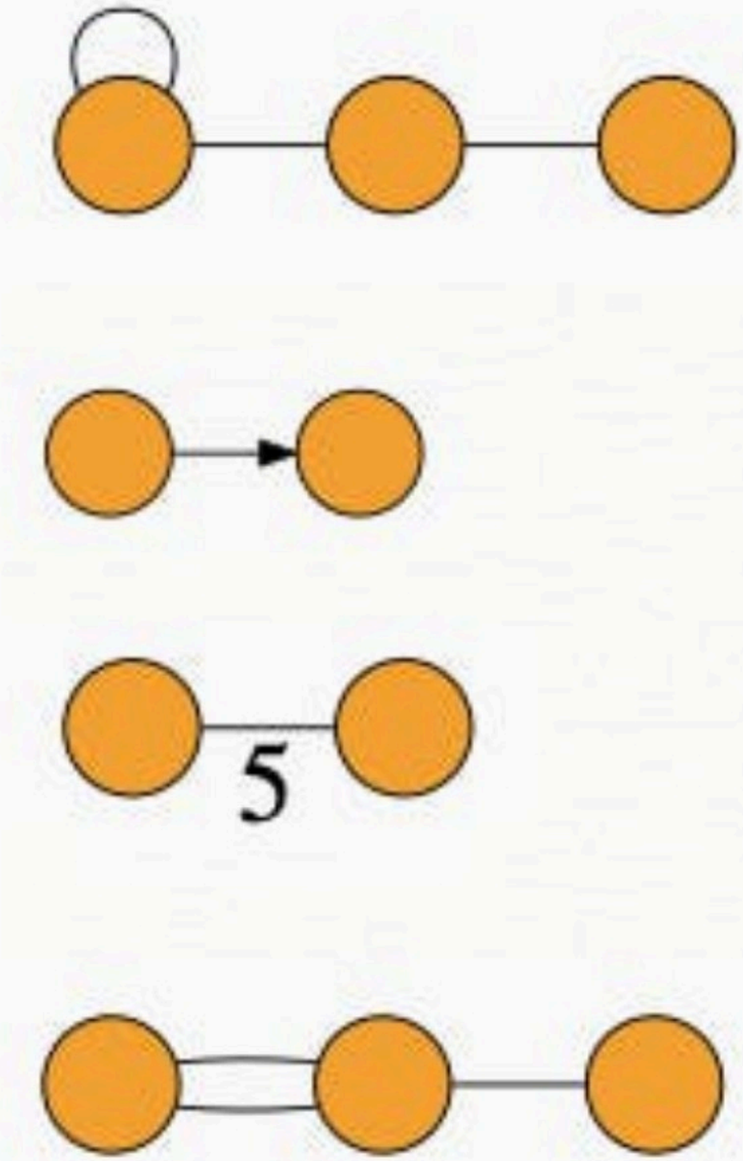
Plan for dagen

- → Definisjon av grafer (masse begreper)
- → Representasjon
- → Traversering av grafer (BFS og DFS)
- → Topologisk sortering

Grafer er...

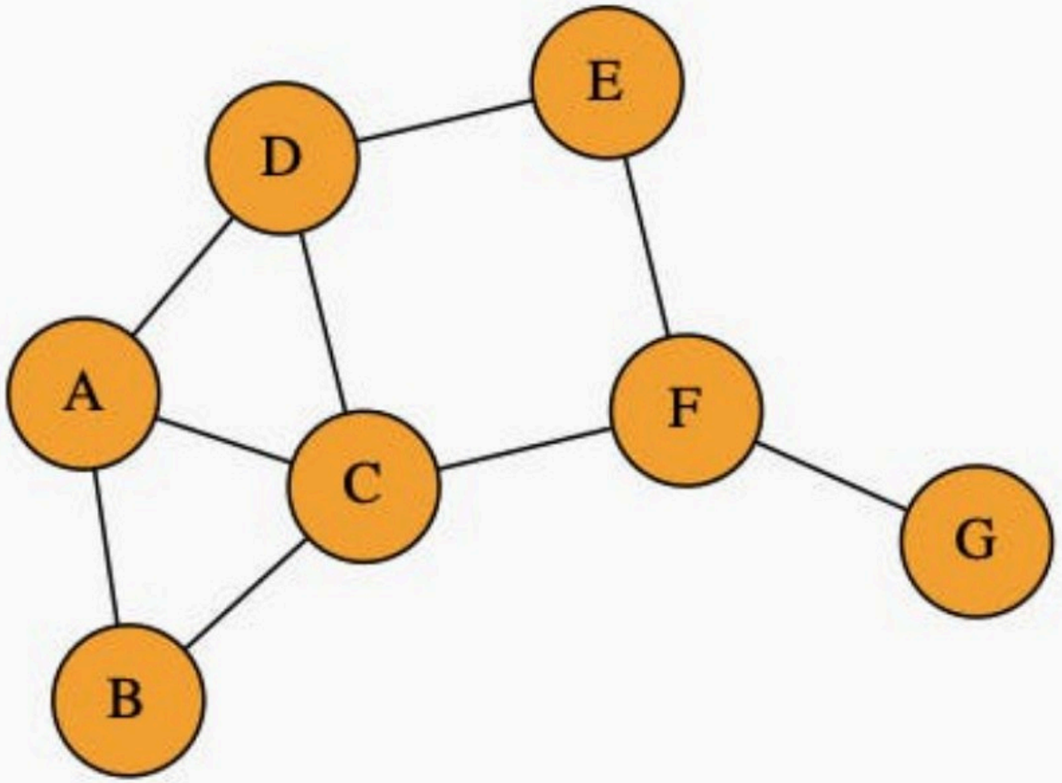
En mengde noder (**V** for vertices), og en mengde kanter (**E** for edges) mellom noder

Alle trær er grafer, men ikke alle grafer er trær!



Begreper, *grafer kan ha/være...*

- (Enkle) løkker
- Parellelle kanter
- Vektet/uvektet
- Rettet/urettet
- *Enkel* graf
- Syklisk/asyklisk



Flere begreper

- Stier (ikke gjenta noder)
- Veier (ikke gjenta kanter)
- Grad (inn- og ut-grad)
- Komplett graf
- Komponenter (*"sub-graf"*)

Hva kan vi beskrive ved hjelp av grafer?

IN2010

Sannsynlighet	relasjoner	kart
Ivennekretser	LinkedIn-forbindelser	relasjoner
data	filtrær	Data med mange relasjoner

Hva kan vi beskrive ved hjelp av grafer?

IN2010

klassehierarki

morgenrutine

nettverk

```python
class Node:
    def __init__(self, data):
        self.data= data
        self._naboer = []

    def leggTilNabo(self, nabo):
        self._naboer.append(nabo)

class Graf:
    def __init__(self):
        self._graf = {}

    def leggTilKant(self, u, v):
        uNode = self._hentNode(u)
        vNode = self._hentNode(v)

        if (self._rettet):
            uNode.leggTilNabo(vNode)
        else:
            uNode.leggTilNabo(vNode)
            vNode.leggTilNabo(uNode)

    def _hentNode(self, data):
        if (data not in self._graf):
            self._graf[data] = Node(data)
        return self._graf[data]
```

Representere grafer

Nabolister, nabomatrise, (objekt-orientert).

Kombinasjon av OO og naboliste
Node1.kanter = [Node2, Node3, Node4]

Prekode: https://github.uio.no/veroninh/IN2010-Gruppe-3/blob/master/uke4/graf_prekode.py

DFS, Dybde-først søk

- Reis så dypt som mulig i en retning, så en annen, osv.
- Kan gjøres med en stack (enklere med rekursjonsstacken)
- O(|V|+|E|)
- *I et tre ville man gått helt ned til løvnoden*

```
def DFS(G,s):
    visited(s) = true                    # Markerer start-noden som besøkt

    for each edge (s,v) do:              # For hver kant ut fra start-noden,
        if visited(v) = false then:      # om kanten leder til en ikke-besøkt node,
            DFS(G,v)                     # kall på DFS med den
    end

# Sørger for alle noder blir besøkt dersom vi har flere komponenter
def DFS-full(G):
    for each vertex v in G do:           # For hver node i grafen,
        if visited(v) = false then:      # om noden ikke er besøkt,
            DFS(G,v)                     # kall på DFS med den
    end
```

Dybde-først søk pseudokode

BFS, Bredde-først søk

- Besøk alle noder lagvis, fra startnoden/sentrum og utover
- Kan bruke en kø, legg til alle noder i neste lag på køen
- O(|V|+|E|)
- *I et tre ville man sett på alle noder med samme dybde, så dybde-1, osv*

```
def BFS(G, s):
    queue = new empty []

    visited(s) = true                    # Markerer start-noden som besøkt
    queue.add(s)                         # og legger den til i køen

    while queue.size() != 0:             # Mens det er noder igjen i køen,
        v = queue.pop(0)                 # fjern neste node

        for each edge (v, w) do:         # For hver kant ut fra noden,
            if visited(w) = false then: # om kanten leder til en ikke-besøkt node,
                visited(w) = true        # så besøker vi den
                queue.add(w)             # og legger den til i køen
            end
        end
    end
```

Bredde-først søk, pseudokode

Topologisk sortering

- Bruker DAG-graf (rettede, asykliske grafer)
- Topologisk sortering gir oss lovlig rekkefølge av "utføring" (eller sier ifra om sykel)
  - Legg alle noder med inngrad 0 på køen (dvs. ingen avhengigheter)
  - Plukk vekk en node fra køen
  - Fjern kantene som peker ut av denne noden
  - Repeter til grafen er tom
- O(|V|+|E|)

```
def TopologiskSortering(G):
    S = new empty Stack
    for each vertex v in G do:          # Legger til alle noder med null inngående kanter
        if inDeg(v) == 0 then:          # (dvs. inDeg=0) på stacken
            S.push(v)
    end

    i = 0                               # Teller for antall elementer vi "utfører"
    output = []
    while S not empty do:               # Mens vi enda har flere ting igjen å "uføre"
        v = S.pop()                     # Plukker ut en node og
        output[i] = v                   # "utfører" den
        i += 1

        for each edge (v,w) in G do:    # Minsker inDeg for alle naboene til v,
            inDeg(w) -= 1
            if inDeg(w) == 0 then:      # legger dem til på stacken om de får inDeg = 0
                S.push(w)
        end
    end

    if i == G.size() then:
        return output
    else:                               # Om vi ikke har utført like mange noder
        return "G has a cycle"          # som det er i grafen, så har vi en syklus
```

Topologisk sortering, pseudokode