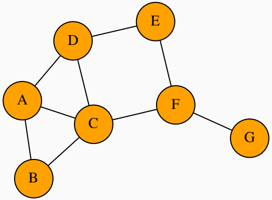
Grafer: Definisjoner og Representasjon

Uke 38, 2020

Institutt for Informatikk

Grafer

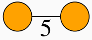
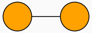
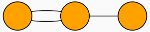
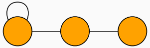
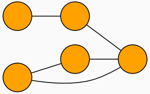


- noder $\{A, B, C, D, E, F, G\}$
- Kanter $\{A, B\}, \{B, C\}, \{C, F\}, \ldots$

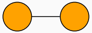
Formelt består en graf av to mengder (V, E)

- 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.

Terminologi

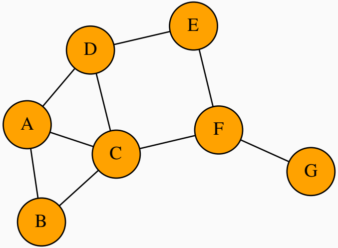
Betegnelse	Forklaring	Eksempel
Vektet/uvektet	Kantene har en verdi/kostnad knyttet til seg	 vs 
Parellelle kanter	Flere enn én kant mellom to noder	
(Enkle) løkker	En kant fra en node til seg selv	
Enkel graf	Ingen løkker, ingen parallelle kanter, urettet, uvektet	

Rettede/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).

Veier og stier

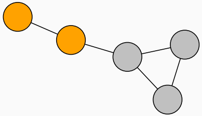


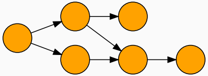
- Sti: En sekvens av noder i grafen forbundet av kanter, der ingen *noder* gjentas. A, B, C, D
- Vei: En sekvens av noder i grafen forbundet av kanter, der ingen *kanter* gjentas. A, B, C, A, D

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

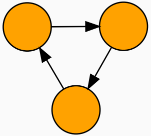
Sykler

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



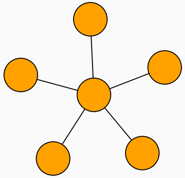


- Grafer uten sykler kalles *asykliske*.
- i rettede grafer: kantene må "peke i riktig retning"



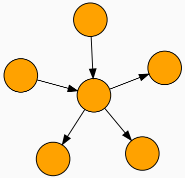
Grad til en node

Graden til en node, $deg(v)$, beskriver hvor mange kanter v er koblet sammen med.



Node i midten har grad 5.

I en rettet graf skiller man mellom inn- og ut-grad til en node.

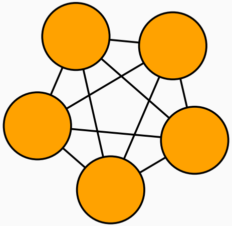


Noden i midten har grad 5: inn-grad 2, ut-grad 3. Alle andre har (inn- eller ut-)grad 1.

Hver kant blir telt to ganger når vi summerer gradene: $deg(v_1) + deg(v_2) + \ldots + deg(v_n) = \sum_{v \in V} deg(v) = 2|E|$

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!)



Representasjon av grafer

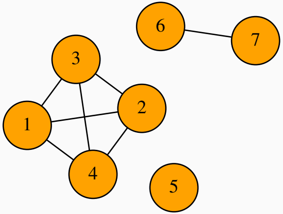
- objektorientering
- nabo-matrise
- nabo-liste

Representasjon av grafer: Objektorientering

- Noder og kanter som objekter
- ligner veldig på den formelle definisjonen
- men veldig lite effektivt...

```
class Vertex {
    String label;
}

class Edge {
    int weight;
    Node from;
    Node to;
}
```



Representasjon av grafer: Nabo-matrise (adjacency matrix)

	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0
2	1	0	1	1	0	0	0
3	1	1	0	1	0	0	0
4	1	1	1	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1
7	0	0	0	0	0	1	0

Fordeler:

- Velegnet for rettede, tette grafer
- å finne eksisterende og legge til nye kanter tar $O(1)$ tid

Ulemper:

- høy minnebruk: $O(|V|^2)$ plass
- å legge til ny node tar $O(|V|)$ tid

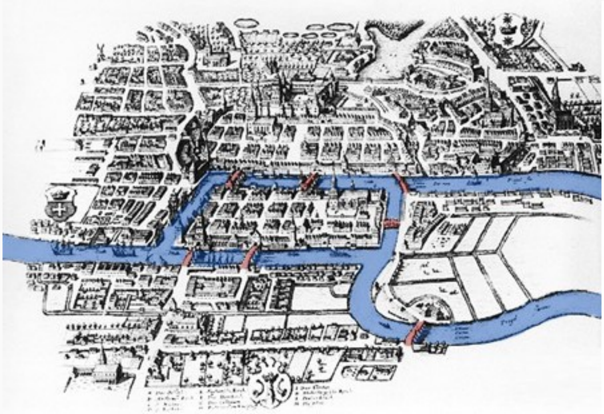
- Typisk implementert som en 2-dimensjonal array, A
- $A[i][j] = 1$ tolkes som at G har en kant fra node i til node j
- Vektede grafer: $A[i][j] = c(i, j)$ tolkes som at G har en kant fra node i til node j med kost $c(i, j)$

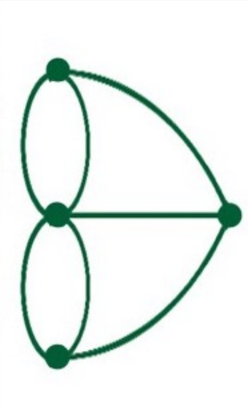
Representasjon av grafer: Nabo-liste (adjacency list)

$$\begin{aligned} nabo(1) &= \{2,3,4\} & nabo(5) &= \emptyset \\ nabo(2) &= \{1,3,4\} & nabo(6) &= \{7\} \\ nabo(3) &= \{1,2,4\} & nabo(7) &= \{6\} \\ nabo(4) &= \{1,2,3\} \end{aligned}$$

- Vi bruker en datastruktur til å lagre noder og sine naboer.
- dette tilsvarer key-value pairs (f.eks. HashMaps), hvor nøkkelen er en node og verdien er en liste/array av sine naboer
- Nabo-lista for node v krever $O(deg(v))$ plass
- G representeres med $O(|V| + |E|)$ plass
- Velegnet for tynne grafer

Klassisk Grafproblem: Eulerkretser

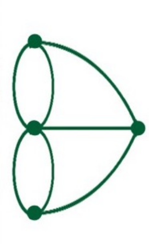




Klassisk Grafproblem: Eulerkretser

- krets: en vei som ikke gjentar kanter som starter og slutter i samme node
- Eulerkrets: For en gitt graf, finnes det en krets som besøker alle kanter nøyaktig en gang?

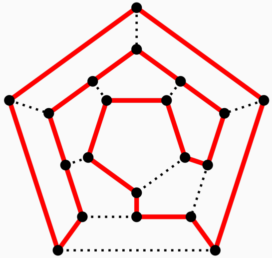
Eulerkrets	
Instans:	En graf G
Spørsmål:	Har G en Eulerkrets?



Klassisk Grafproblem: Hamiltonsykel

- Vil finne en sykel som besøker alle noder nøyaktig en gang

Hamiltonsykel	
Instans:	En graf G
Spørsmål:	Har G en Hamiltonsykel?



To klassiske grafproblemer

Eulerkrets	
Instans:	En graf G
Spørsmål:	Har G en Eulerkrets?

vs.

Hamiltonsykel	
Instans:	En graf G
Spørsmål:	Har G en Hamiltonsykel?

Et problem er lett å løse, det andre (virker) vanskelig (!)