Grafer

I. Graf

definisjon
terminologi
noen eksempler på graf-problemer

II. Graf ADT og Implementasjon

Kant-Liste og Nabo-Liste
Nabo-Matrise

III. Graf Traversering

DFS
BFS

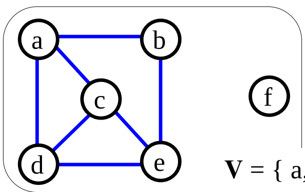
IV. Rettede Grafer (diGraphs)

V. Vektede Grafer

Kap. 9 (kursorisk 9.4.5)
Kap.10 (untatt 10.2.2–10.2.4, 10.3)

En graf G

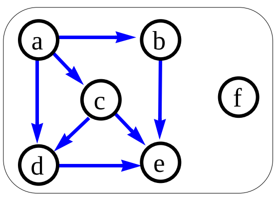
er gitt ved to mengder V og E; G=(V,E)
V er noder (boka: vertices)
E er kanter (boka: edges)
en kant e ε E er
et (uordnet) par {u,v} av noder u ε V og v ε V



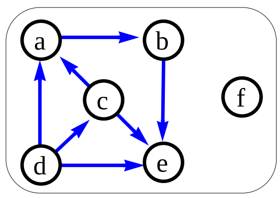
V = { a, b, c, d, e, f }
E = { (a,b), (a,c), (a,d), (b,e), (c,d), (c,e), (d,e) }
= { (b,a), (c,a), (d,a), (e,b), (d,c), (e,c), (e,d) }

En **ikke-rettet graf** kan sees på som en (rettet) graf der relasjonen E er **symmetrisk** u → v ε E *hviss* v → u ε E.

En **rettet graf** (diGraf)
V av noder
E av kanter
der en kant e ε E er
et ordnet par (u,v), u ε V, v ε V, dvs u → v



E = { (a,b), (a,c), (a,d), (b,e), (c,d), (c,e), (d,e) }



E = { (a,b), **(c,a)**, **(d,s)**, (b,e), **(d,c)**, (c,e), (d,e) }

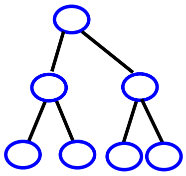
En (rettet) **graf** er gitt ved
en mengde V av noder
en **binær relasjon** E $\subseteq V \times V$

Anvendelser ...

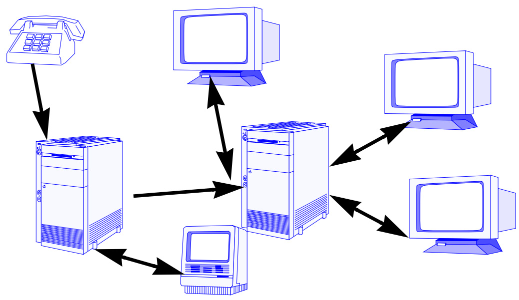
I. BINÆRE RELASJONER

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow$

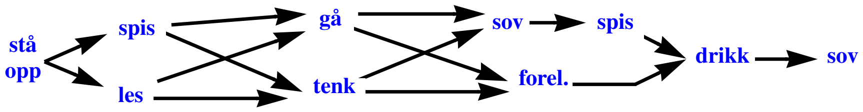
II. TRÆR



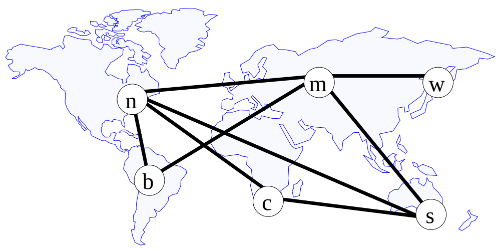
III. NETTVERK

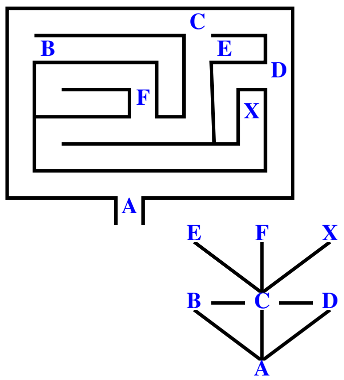


IV. PLANLEGGING



V. FORBINDELSER





. . .

Graf-terminologi

nabonoder: 2 noder som er forbundet med en kant — a-c, c-e

gradtall til en node: antall nabonoder (kanter), angitt ved deg(v) — deg(c)=3, deg(f)=0

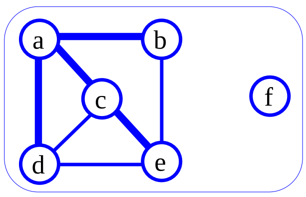
$$\sum_{v \in V} deg(v) = 2(\#\ \text{kanter})$$

inngrad/utgrad: antall innkommende/utgående kanter (rettede grafer)

sti: en sekvens $n_1, n_2 \ldots n_k$ av noder slik at $(n_i, n_{i+1}) \varepsilon E$ — acaca, bec, abedc, edce

enkel sti: en sti hvor ingen node forekommer 2 ganger

sykel: en sti hvor $n_1 = n_k$

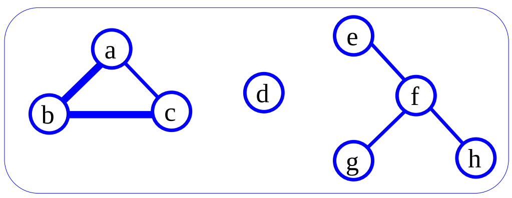


sammenhengende graf: graf hvor det finnes en sti mellom alle par av noder

delgraf: en delmengde av V og E som er en graf

sammenhengende komponent: en maksimal sammenhengende delgraf

komplett graf: om det for hvert par av noder u, v ε V, finnes en kant (u,v) ε E



(ikke-rotet) **tre**: sammenhengende graf uten sykler

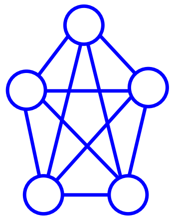
utspennende tre for en graf G: en delgraf av G som er et tre og inneholder alle G's noder

skog: samling av trær

DAG: rettet graf uten sykler (directed acyclic graph)

1. Antall noder og kanter

n = # noder i grafen, ***m*** = # kanter i grafen



n =5
k =10

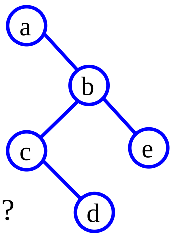
- G er **komplett hviss** hver node har (n –1) naboer

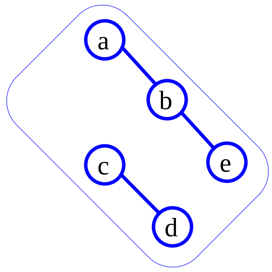
dvs. $m = 1/2 * \sum_{v \in V} deg(v) = 1/2 * n * (n-1)$

- G **ikke komplett hviss** $m < 1/2 * n * (n-1)$

- **Hvis** G **er et tre så er** $m = n - 1$

n =5
k =4



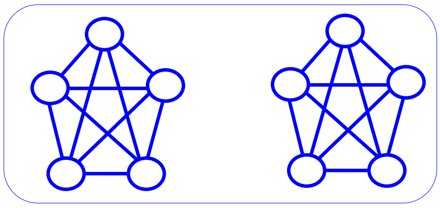


- dette er det minste antall kanter som kan gi en sammenhengende graf på n noder. Bevis?

- fjerning av en vilkårlig kant, gjør treet usammenhengende. Bevis?

- G trenger ikke å være et tre selv om $m = n - 1$ (G kan være usammenhengende)

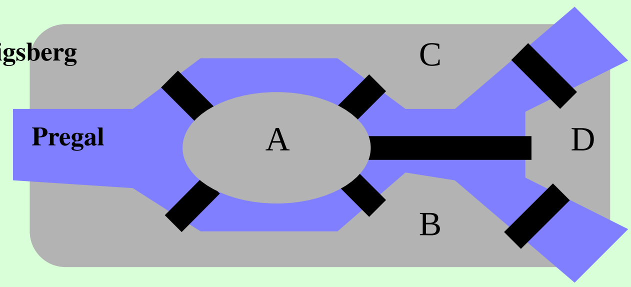
- **Hvis** $m < n - 1$ **så er ikke** G **sammenhengende**
men ikke omvendt. Hvorfor ikke?



n =10
k =20 > 9

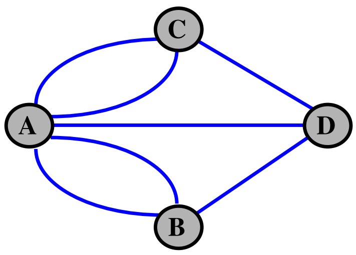
Eulers tur i en vilkårlig graf

Euler (1736):
Kan jeg under min aftentur krysse hver bro nøyaktig én gang ?



Euler (1736) :
Nei – og jeg kan bevise det v.hj.a. grafer !

Bruk multigrafer (som under), eller la broer være noder med gradtall 2.



Eulers tur : en sykel som traverserer hver kant i grafen nøyaktig én gang

Eulers teorem : En (sammenhengende) graf G har en Eulers tur hvis og bare hvis G har 0 eller 2 noder hvor gradtallet er et oddetall. Hvorfor? 'Lett' å finne:**O(n+m)**

Hamiltonsk sykel : en sykel som traverserer hver node nøyaktig en gang. Vanskelig å finne: **O(n!)**

... **...**

Graf ADT *(ikke-rettet utsnitt)*

package jdsl.core.api;

```
public interface PositionalContainer
                    extends Container {
// size(); isEmpty(); elements();
  Enumeration positions();
  void swap(Position p, Position q);
  object replace(Position p, Object o);
}
```

```
public interface Position {
  Object element();
  Container container();
  void setElement(Object o);
}
```

```
public interface InspectableGraph
                    extends PositionalContainer {
   int numVertices();
   int numEdges();
   Enumeration vertices();
   Enumeration edges();

   Vertex[] endVertices(Edge e)
   Vertex opposite(Vertex v, Edge e);

   int degree(Vertex v);
   Enumeration adjacentVertices(Vertex v);
   Enumeration incidentEdges(Vertex v);
   ...
}
```

```
public interface Edge
      extends Position { }
```

```
public interface Vertex
      extends Position { }
```

```
public interface Graph extends InspectableGraph {
  Vertex insertVertex(Object o);
  Edge insertEdge(Vertex u, Vertex v, Object o);
  Object removeEdge(Edge e);
  Object removeVertex(Vertex v); !!!
  ...
}
```


Implementasjoner av Graph

n : antall noder **m** : antall kanter

kompleksitet **operasjon**	**Kant-Liste**	**Nabo-Liste**	**Nabo-Matrise**
size, isEmpty	1	1	1
newContainer	1	1	1
elements, positions	n+m	n+m	n+m
replace, swap	1	1	1
numVertices(), numEdges()	1	1	1
vertices() / edges()	n / m	n / m	n / m
degree(v)	1	1	1
endVertices(e), opposite(v,e)	1	1	1
adjacentVertices(v), incidentEdges(v)	m	deg v	n
insertVertex(o)	1	1	n^2
removeVertex(v)	m	deg v	n^2
insertEdge(v,u,o)	1	1	1
removeEdge(e)	1	1	1
areAdjacent(v,u)	m	deg u|v	1