729G46 Informationsteknologi och programmering

Tema 2, Föreläsning 2.3

Johan Falkenjack, johan.falkenjack@liu.se

LINKÖPINGS UNIVERSITET

Frågor i PeerWise

- Använd rätt ämnestagg i PeerWise
- Används av kursdeltagare för att söka ut frågor för specifikt seminarium
- Används av lärare för att söka ut frågor och svar för specifikt seminarium

- Begreppsseminarium 2: Filer
- Begreppsseminarium 3: Internet

Föreläsningsöversikt

- Grafer och grafterminologi
- Tupler
- Träd

Grafer

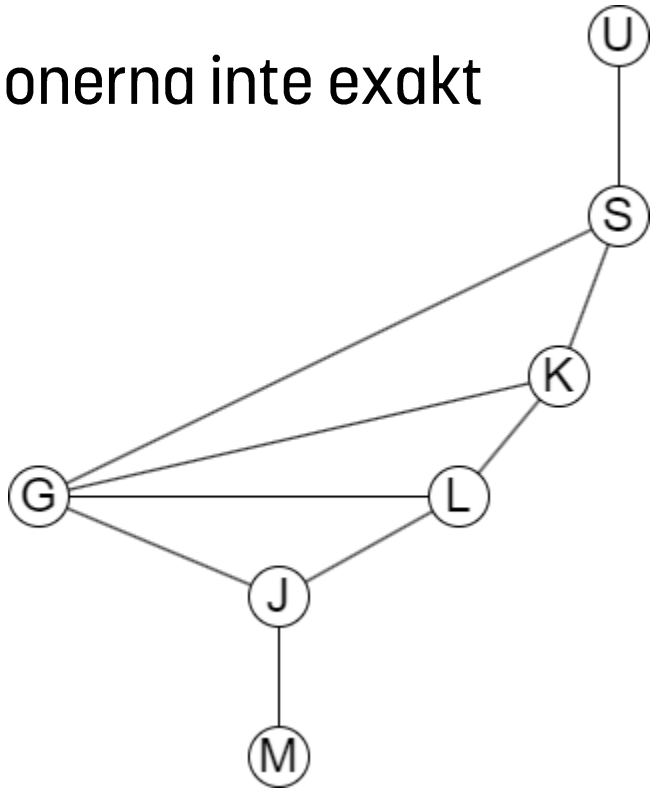
Grafer

- Grafer används för att representera struktur hos en mängd (information/data), hur element är relaterade till varandra.
- Exempel på element vi skulle kunna tänka oss ha en strukturell komponent:
- $S = \{Linköping, Göteborg, Jönköping, Malmö, Katrineholm, Stockholm, Umeå\}$

- $M = \{Ada, Bert, Cilla, Dan, Evy\}$

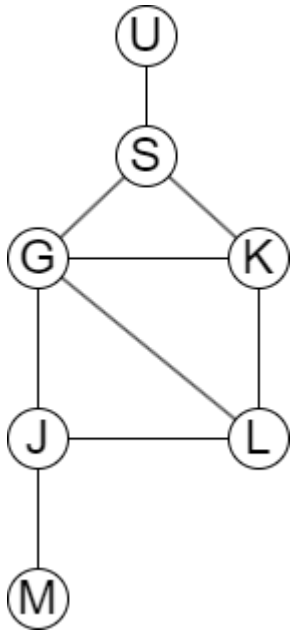
Exempel

- Tågförbindelser mellan olika städer (Notera att positionerna inte exakt motsvarar var städerna ligger geografiskt)



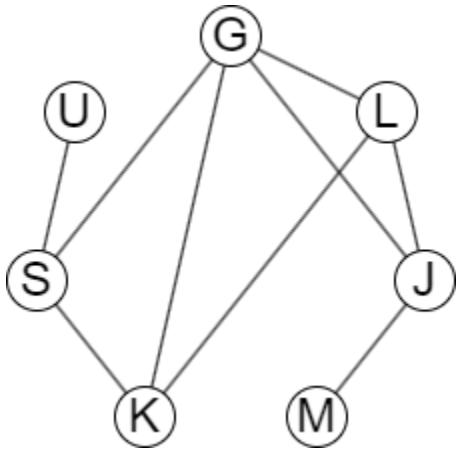
Exempel

- Tågförbindelser mellan olika städer (Notera att positionerna inte exakt motsvarar var städerna ligger geografiskt)



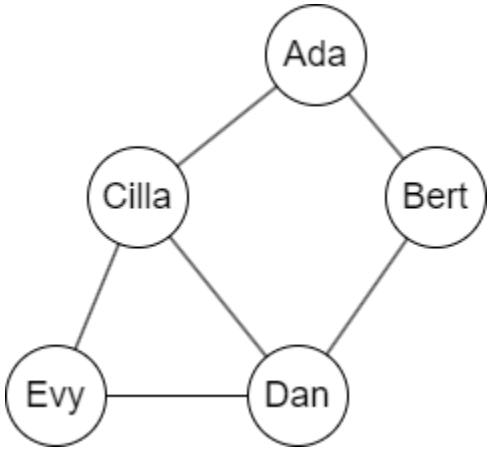
Exempel

- Tågförbindelser mellan olika städer (Notera att positionerna inte exakt motsvarar var städerna ligger geografiskt)



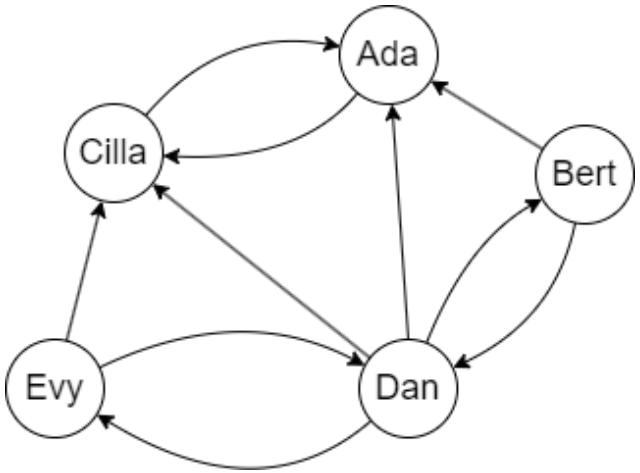
Exempel

- Socialt nätverk där användare kan bli vänner med varandra



Exempel

- Socialt nätverk där användare kan följa varandra



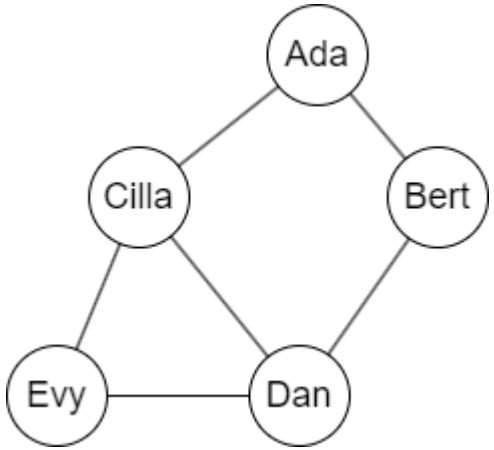
Grafer

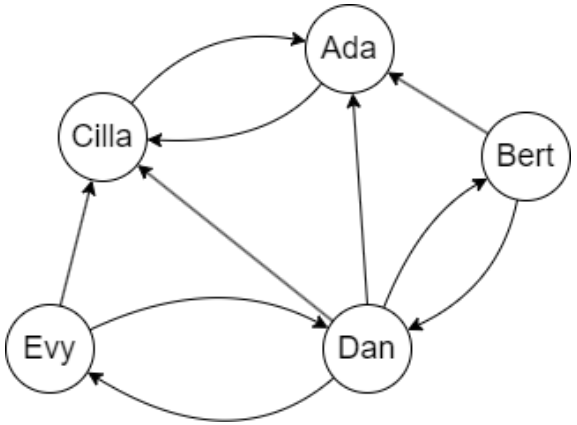
- Grafer har

  noder/hörn (eng. vertex (pl. vertices) / node) och

  bågar/kanter (eng. arc/edge)

- Grafer kan vara riktade eller oriktade





Grafer på formen $G = (V, E)$

Tupler

- Mängd: samling av godtyckliga objekt som kallas element, utan inbördes ordning. Skrivs inom klammerparenteser.

  T.ex. $\{a, b, c, d\}$, där $\{a, b, c, d\} = \{b, a, d, c\}$

- Tupel: ordnat par med inbördes ordning. Skrivs inom bågparenteser

  T.ex. (a, b), där $(a, b) \neq (b, a)$

- n-Tupel: Tupel av längden n

  T.ex. en 3-tupel (f, b, c), även kallad för en *trippel*

Matematisk representation av en oriktad graf

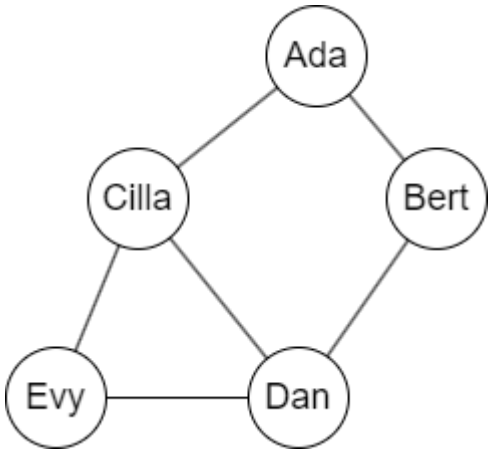
- En graf består av en mängd noder, V, och en mängd bågar E.

  För exempelgrafen är $V = \{a, b, c, d, e\}$

- En oriktad båge har ingen specifik start- eller slutnod, så vi representerar en oriktad båge som en mängd, t.ex. $\{a,b\}$

  För exempelgrafen är $E = \{\{a, b\}, \{a, c\}, \{b, d\}, \{c, d\}, \{c, e\}, \{d, e\}\}$

- Vi kan då definiera grafen G som: $G = (V, E)$
- Grafen är en tupel, eftersom ordningen mellan V och E spelar roll. Den första mängden är noderna, den andra mängden är bågarna.



•För exempelgrafen är E={$(b,a),(b,d),(c,a),(d,c),(d,e),(e,c)$}

Matematisk representation av en riktad graf

- En graf består av en mängd noder, V, och en mängd bågar E.

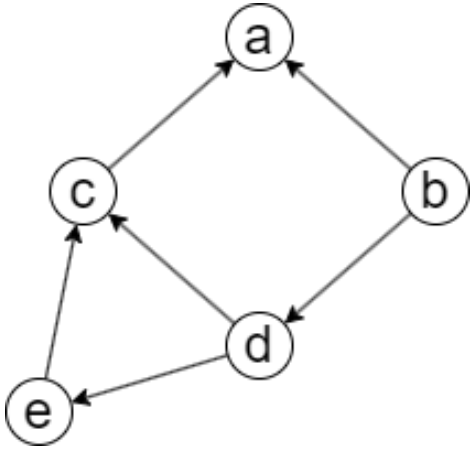
  För exempelgrafen är $V = \{a, b, c, d, e\}$

- En riktad båge har en start- och en slutnod, så vi representerar en riktad båge som en tupel, t.ex. (b, a)

  För exempelgrafen är $E = \{(b, a), (b, d), (c, a), (d, c), (d, e), (e, c)\}$

- Vi kan då definiera grafen G som: $G = (V, E)$

- Grafen är en tupel, eftersom ordningen mellan V och E spelar roll. Den första mängden är noderna, den andra mängden är bågarna.



Delgrafer

- En delgraf, $G'=(V',E')$ är en delgraf till G om $V' \subseteq V$ och $E' \subseteq E$.

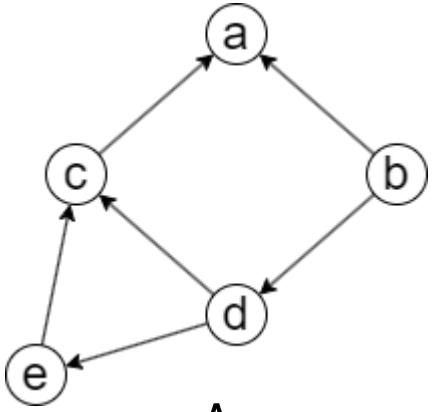
  $V_A = \{a, b, c, d, e\}$

  $E_A = \{(b, a), (b, d), (c, a), (d, c), (d, e), (e, c)\}$

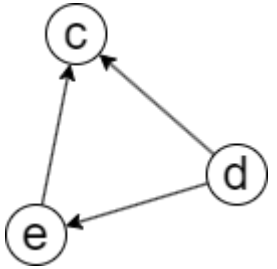
  $V_B = \{c, d, e\}$

  $E_B = \{(d, c), (d, e), (e, c)\}$

  $V_B \subseteq V_A$ och $E_B \subseteq E_A$, alltså är B en delgraf till A



A



B

Delgrafer

- En delgraf, $G'=(V',E')$ är en delgraf till G om $V' \subseteq V$ och $E' \subseteq E$.

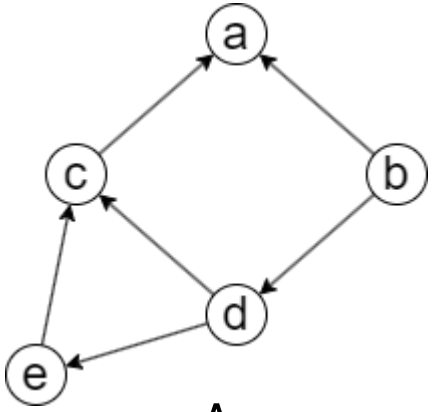
  $V_A = \{a, b, c, d, e\}$

  $E_A = \{(b, a), (b, d), (c, a), (d, c), (d, e), (e, c)\}$

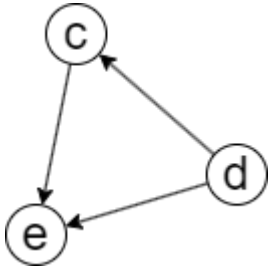
  $V_C = \{c, d, e\}$

  $E_C = \{(d, c), (d, e), (c, e)\}$

  $V_C \subseteq V_A$ och $E_C \nsubseteq E_A$, alltså är C en delgraf till A



A



C

Noder och valens

- Noder som har en båge mellan sig kallas grannar (eng. *neighbors*).
- En nods grad eller valens (eng. *degree* eller *valency*) är antalet bågar som ansluter till noden.

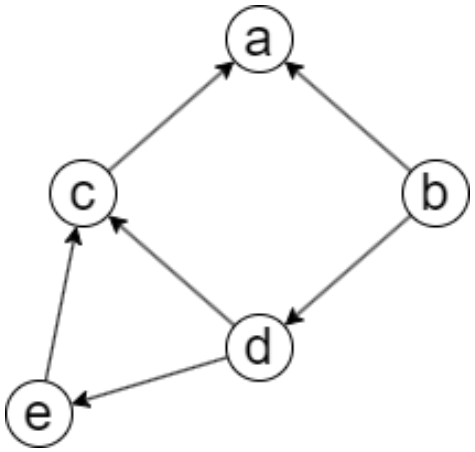
  Om bågarna en nod ingår i är riktade kan man prata om en nods **in- och utvalens** eller **in- och utgrad**.

- Exempel:

  Noden d har gradtalet 3, mer specifikt,

  Ingrad 1

  Utgrad 2

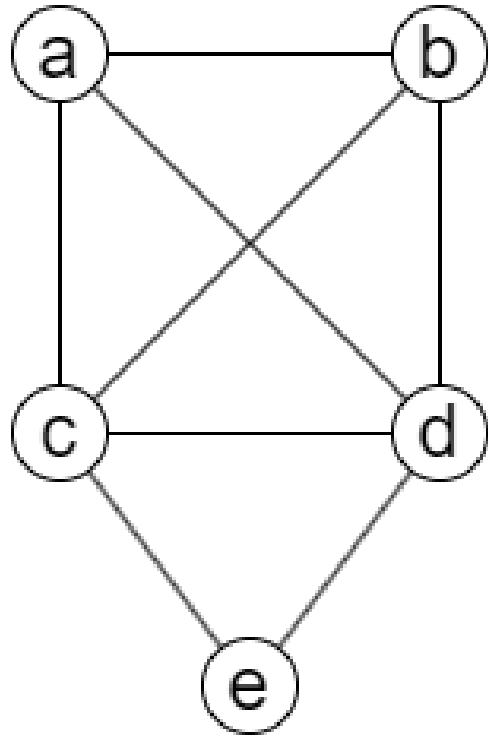


Vandring, väg, stig, krets, cykel

- vandring (*walk*): en följd av noder i en graf, $(v_1, v_2, \ldots v_k)$ där det mellan varje v_i och $v_i + 1$ finns en båge.
- väg (*trail*): en vandring som inte använder samma båge mer än en gång
- stig (*path*), är en väg som inte besöker samma nod två gånger
- En sammanhängande (*connected*) graf är en graf där det finns en vandring mellan varje par av noder.
- En krets (*circuit*) eller en sluten väg är en väg $(v_1, v_2, \ldots v_k)$ där $v_1 = v_k$, dvs som börjar och slutar i samma nod
- En cykel (*cycle*) eller en sluten stig är en stig $(v_1, v_2, \ldots v_k)$ där $v_1 = v_k$, dvs som börjar och slutar i samma nod

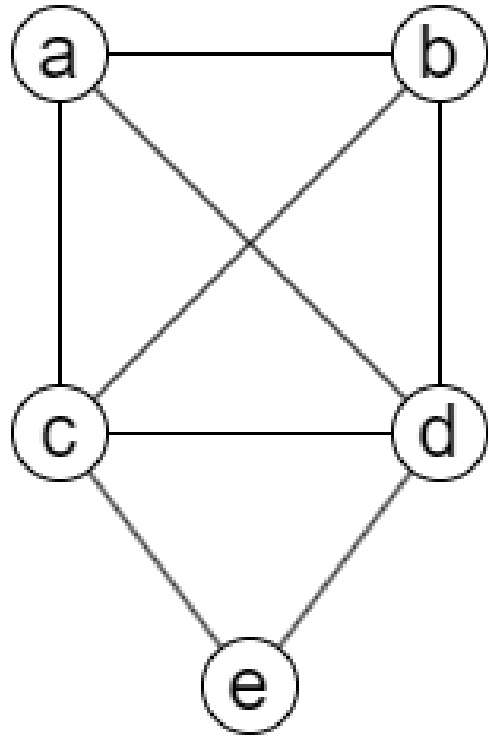
Vandring, väg, stig, krets, cykel

- (a, b, c, d, b, a) är en vandring, men
  inte en väg eftersom den använder bågen $\{a, b\}$ mer än en gång
  inte en stig eftersom den besöker t.ex. noden b flera gånger
  varken en krets eller cykel



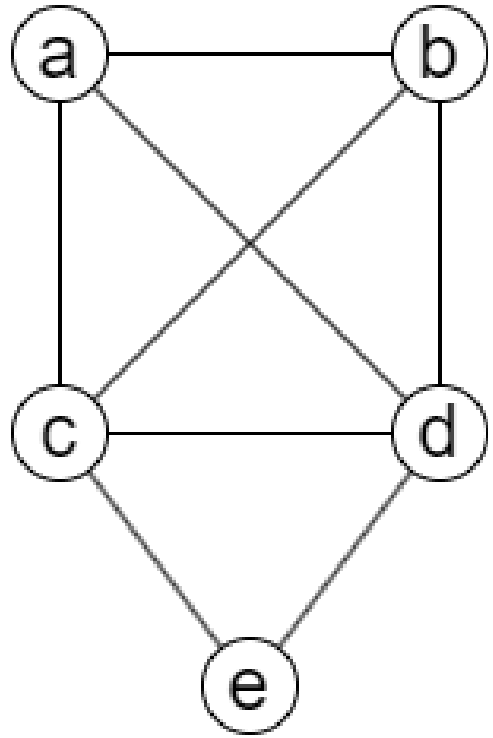
Vandring, väg, stig, krets, cykel

- (a, b, c, a, d) är en vandring och en väg, men
  inte en stig eftersom noden a finns med mer än en gång.



Vandring, väg, stig, krets, cykel

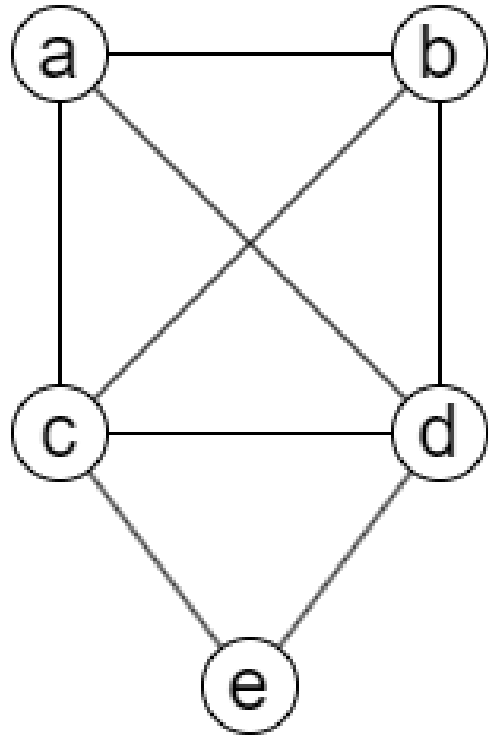
- (a, b, c) är en vandring, en väg och en stig.



Vandring, väg, stig, krets, cykel

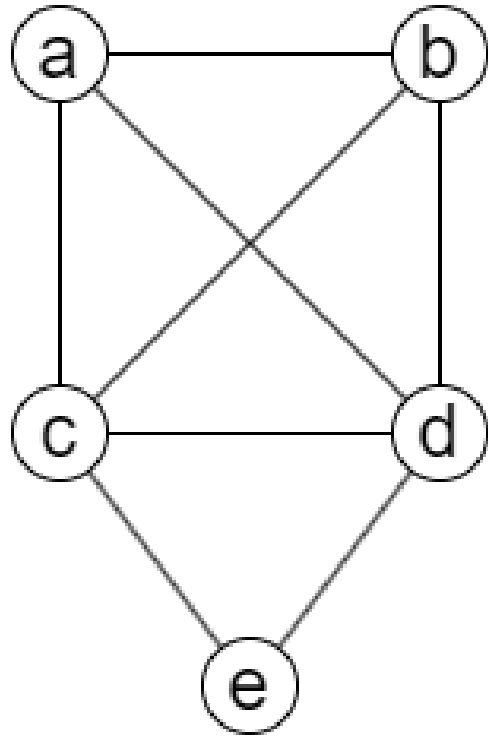
- (a, b, c, d, e, c, a) är en krets men

  inte en cykel eftersom noden c besöks mer än en gång



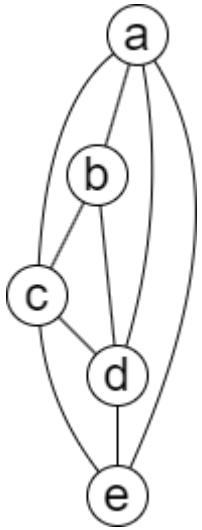
Vandring, väg, stig, krets, cykel

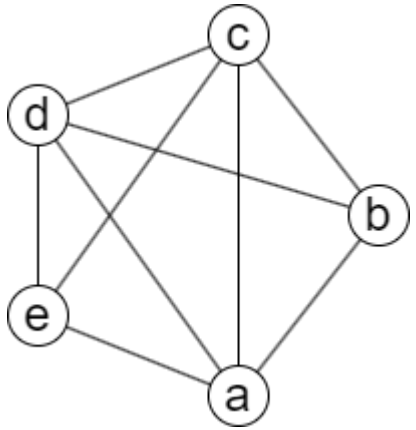
- (a, b, c, d, a) är en krets och en cykel

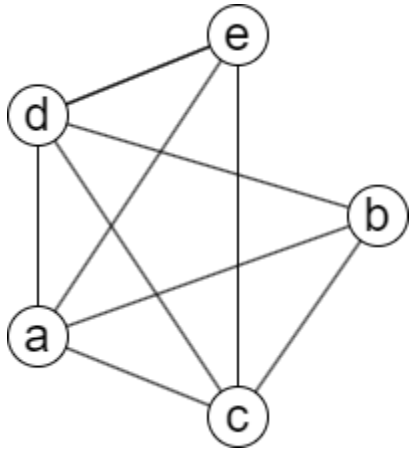


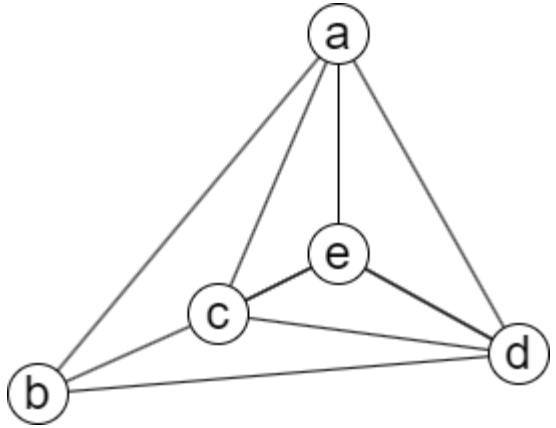
Isomorfi

- Två ritade grafer är isomorfa om strukturen är den samma men graferna är ritade på olika sätt



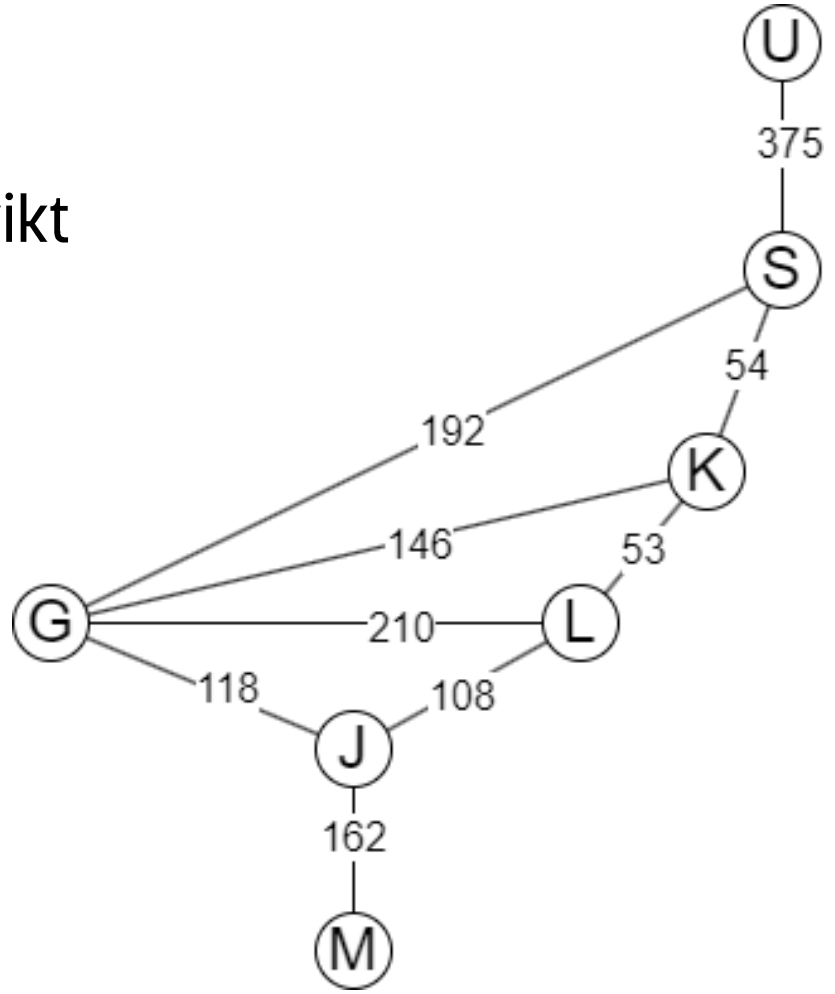






Viktade grafer

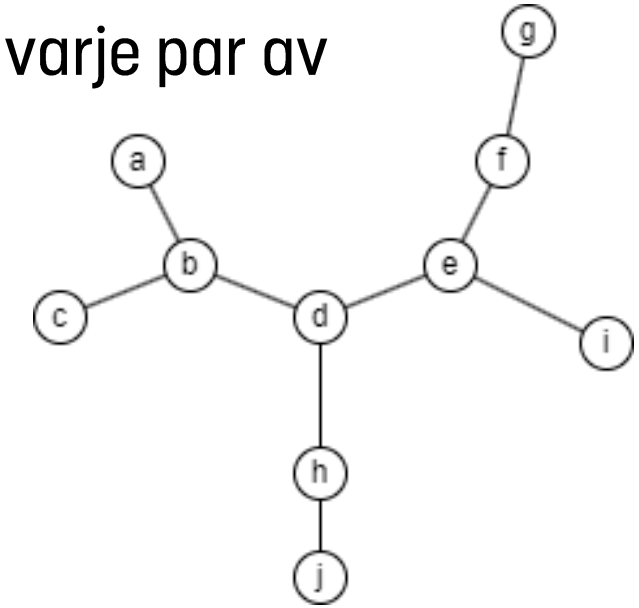
- I en viktad graf har man tilldelat varje båge en vikt
- Exempel: Restid mellan olika städer



Träd - ett specialfall av grafer

Träd

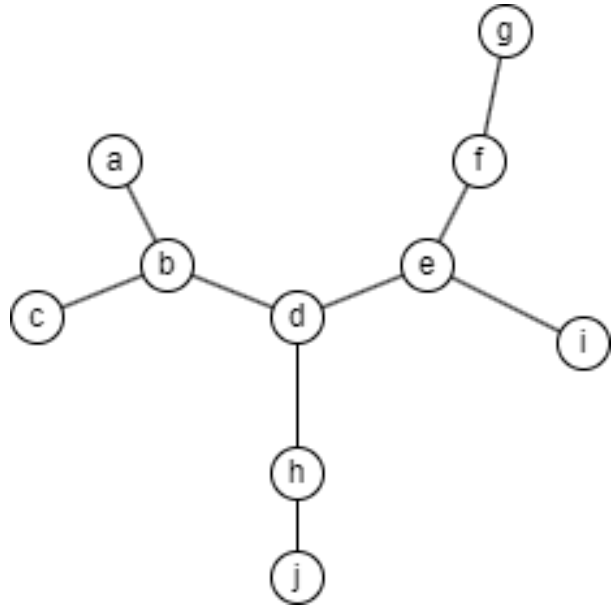
- En graf är ett träd om grafen är
  - Sammanhängande
  - Acyklisk (utan cykler)
- Man kan också beskriva ett träd som en graf där det mellan varje par av noder finns en unik, enkel väg
- Noder med grad 1 kallas för löv
- Träd har alltid $|V| - 1$ bågar



Träd - egenskaper

- Om man lägger till en båge till ett träd får man en cyklisk graf

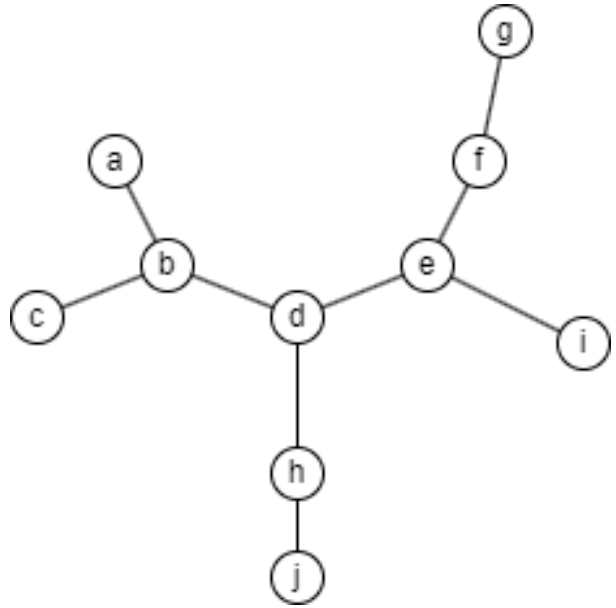
  Ett träd är sammanhängande, dvs det finns en båge mellan varje par av noder. Om vi har ett träd med noderna v och w och lägger till den nya bågen $\{v, w\}$ kommer det nu finnas två vägar mellan v och w. Vi kan vandra från v till w med den ursprungliga vägen, för att sedan gå från w till v med den nya bågen. Vi har en cykel.



Träd - egenskaper

- Om man tar bort en båge från ett träd får man en osammanhängande graf

  Ett träd är sammanhängande och har inga cykler, dvs det finns exakt en väg mellan varje par av noder. Om vi har bågen $\{v, w\}$ vet vi att det är den enda vägen mellan v och w. Om vi tar bort $\{v, w\}$ försvinner alltså den enda vägen mellan v och w och grafen blir osammanhängande.



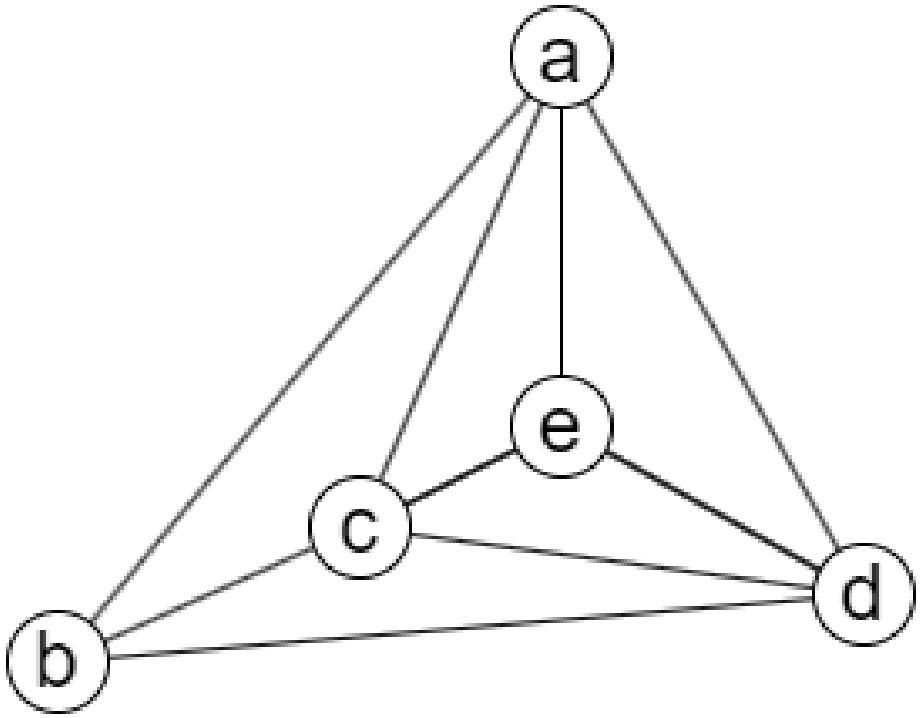
Spännande (även uppspännande) träd

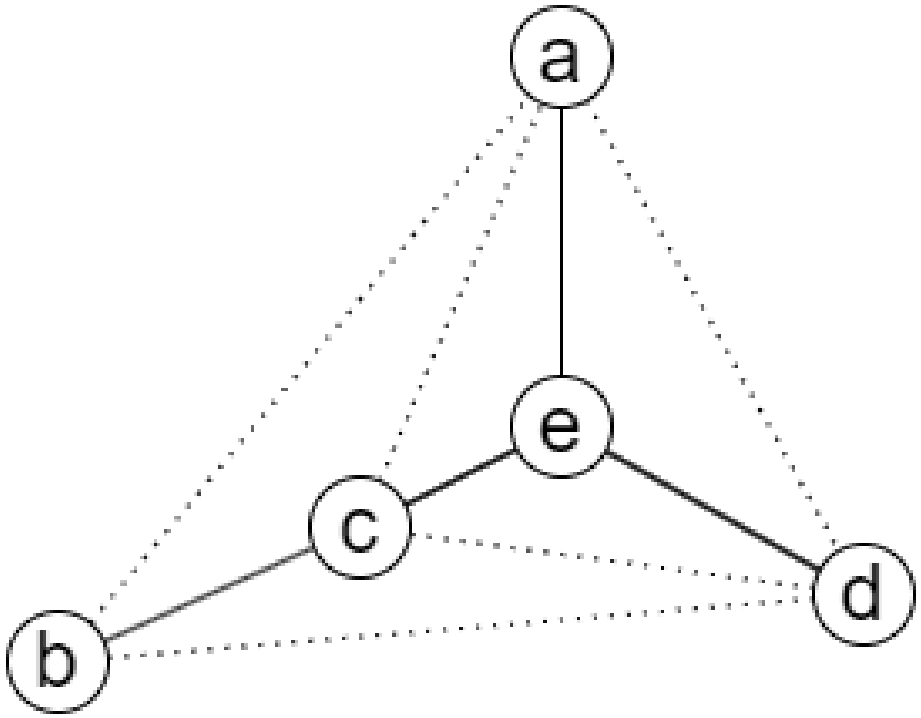
- Om $G = (V, E)$ är en sammanhängande graf, så kan vi ta fram ett spännande träd, G' för G.
- G' är ett spännande träd för G om

  $G' = (V, E')$ där $E' \subseteq E$

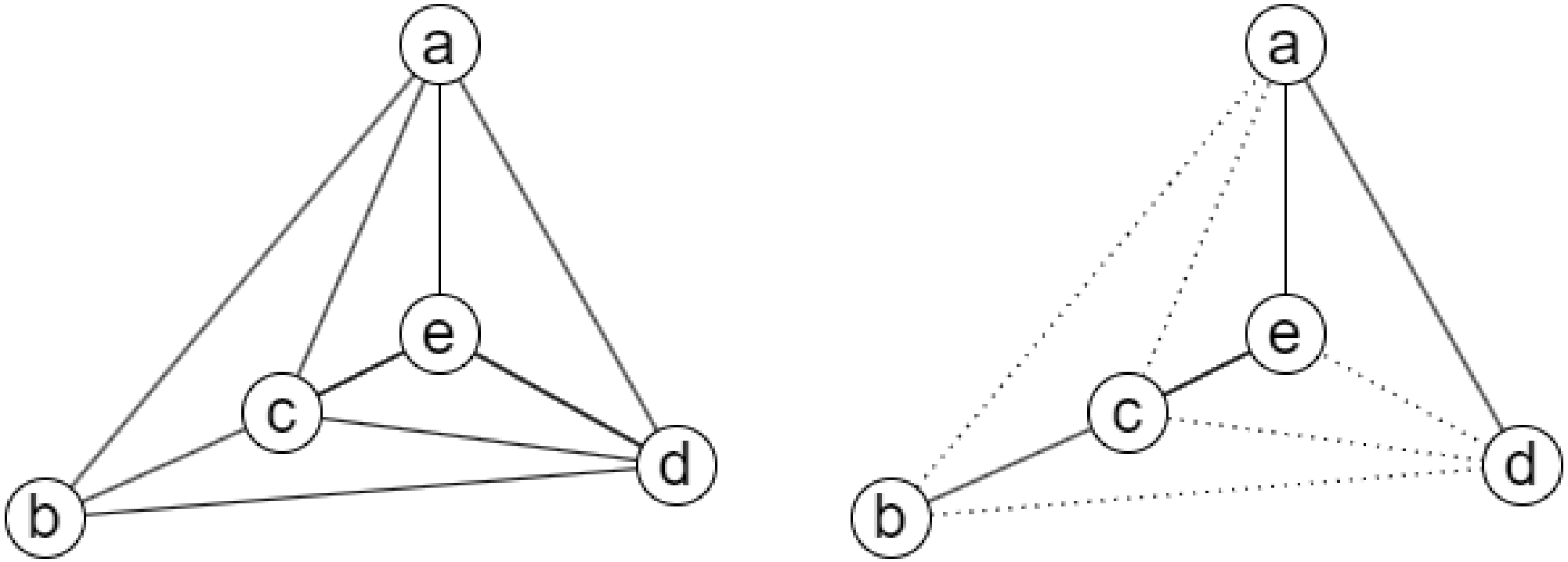
  G' är ett träd
- Ett spännande träd för grafen G innehåller alltså alla noderna från G, men inte nödvändigtvis alla bågar.
- Med en grafs spännande träd har vi exakt en väg mellan varje par av noder eftersom ett träd är sammanhängande och inte innehåller några cykler

Exempel, spännande träd





Exempel, spännande träd

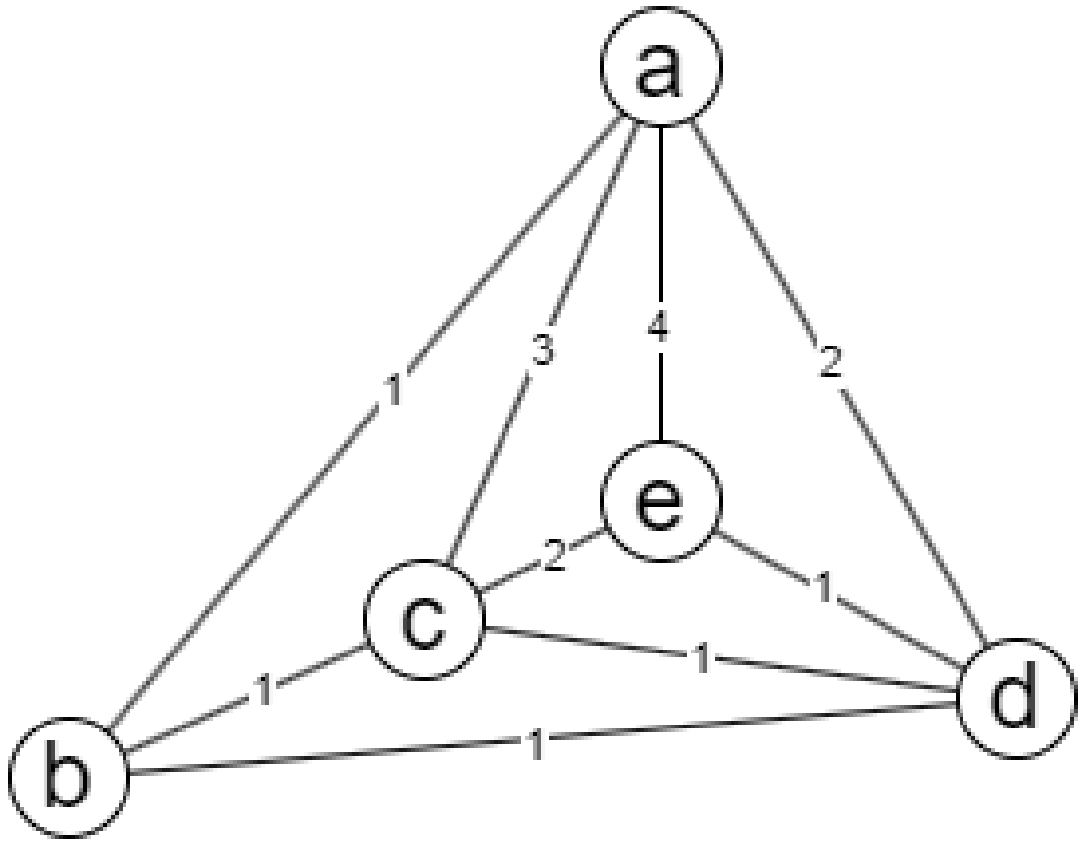


Minimalt uppspännande träd för en (viktad) graf

- Det uppspännande träd som har den *minsta viktsumman* (summan av alla bågars vikter i en graf)
- Kruskals algoritm för att ta fram ett minimalt uppspännande träd
  1. Sortera bågarna efter ökande vikt
  2. Välj en ny båge med lägst vikt
  3. Kontrollera att bågen inte introducerar en cykel
  4. Om den inte introducerar en cykel, lägg till den till det minimalt uppspännande trädet
  5. Upprepa från Steg 2 tills vi har $|V| - 1$ stycken bågar i trädet

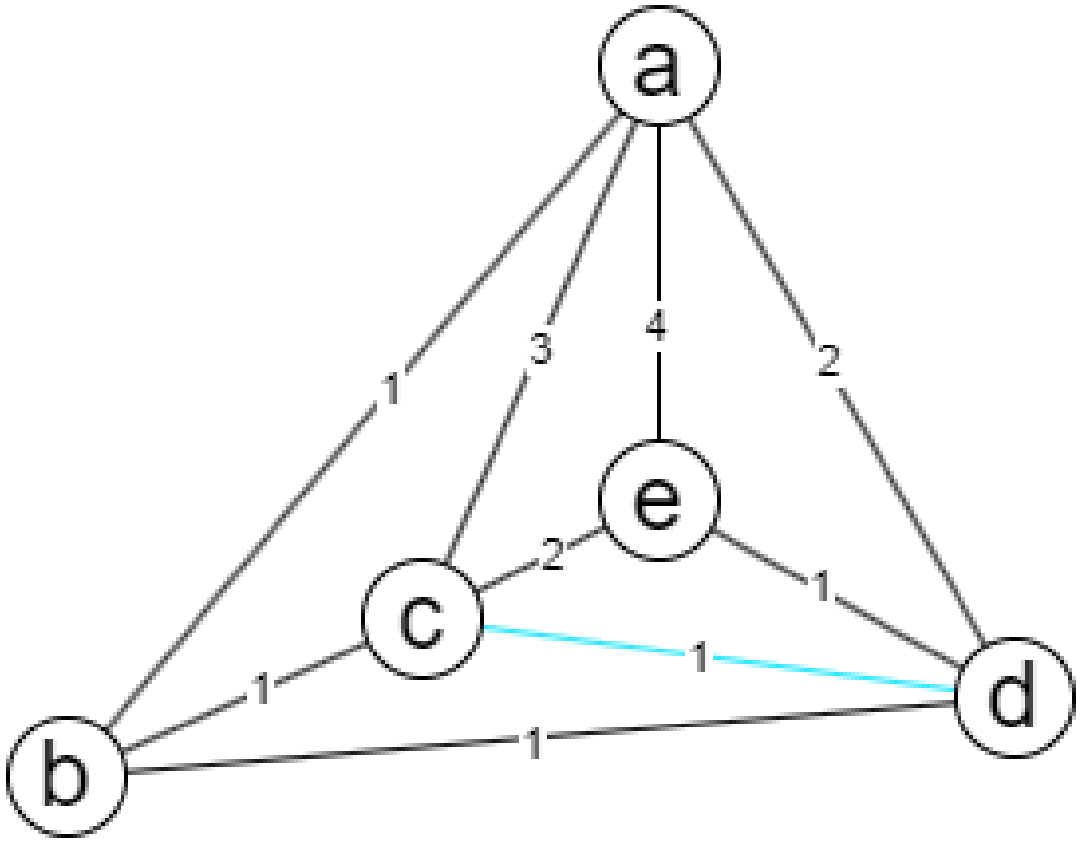
Tillämpning av Kruskals algoritm

- Mål: 5-1=4 bågar



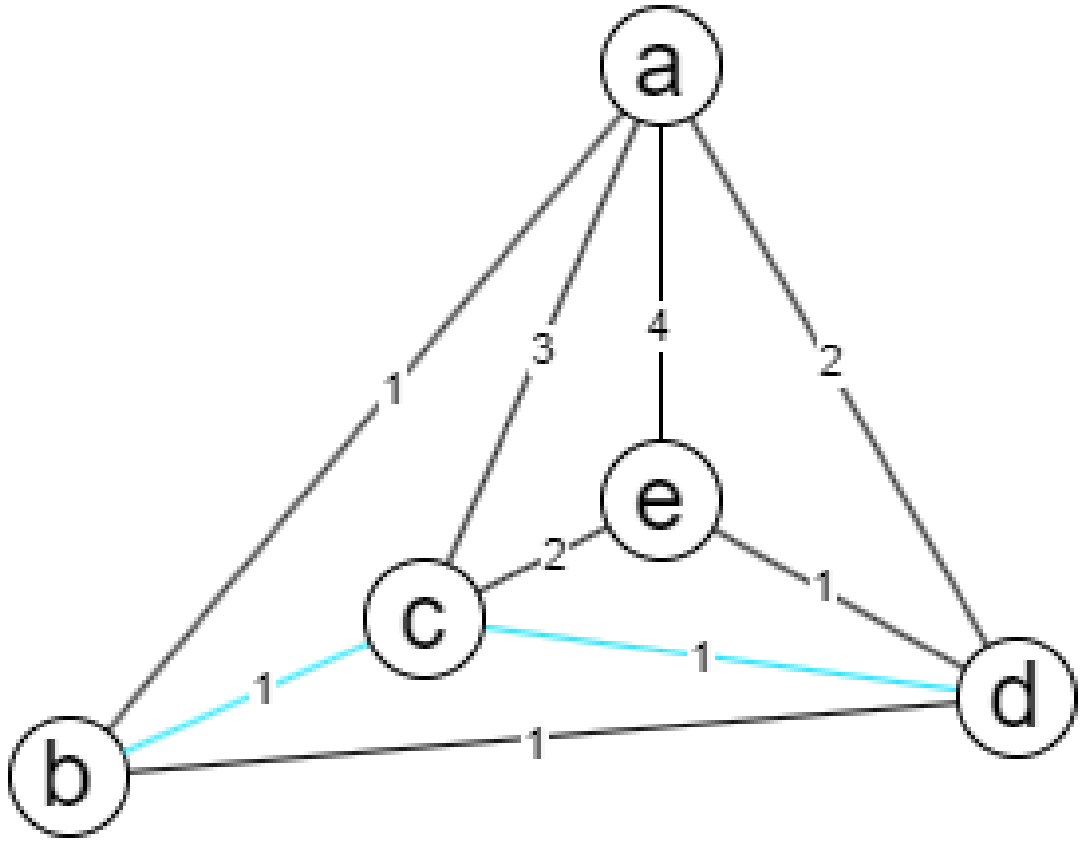
Tillämpning av Kruskals algoritm

- Mål: 5-1=4 bågar
- 1 båge



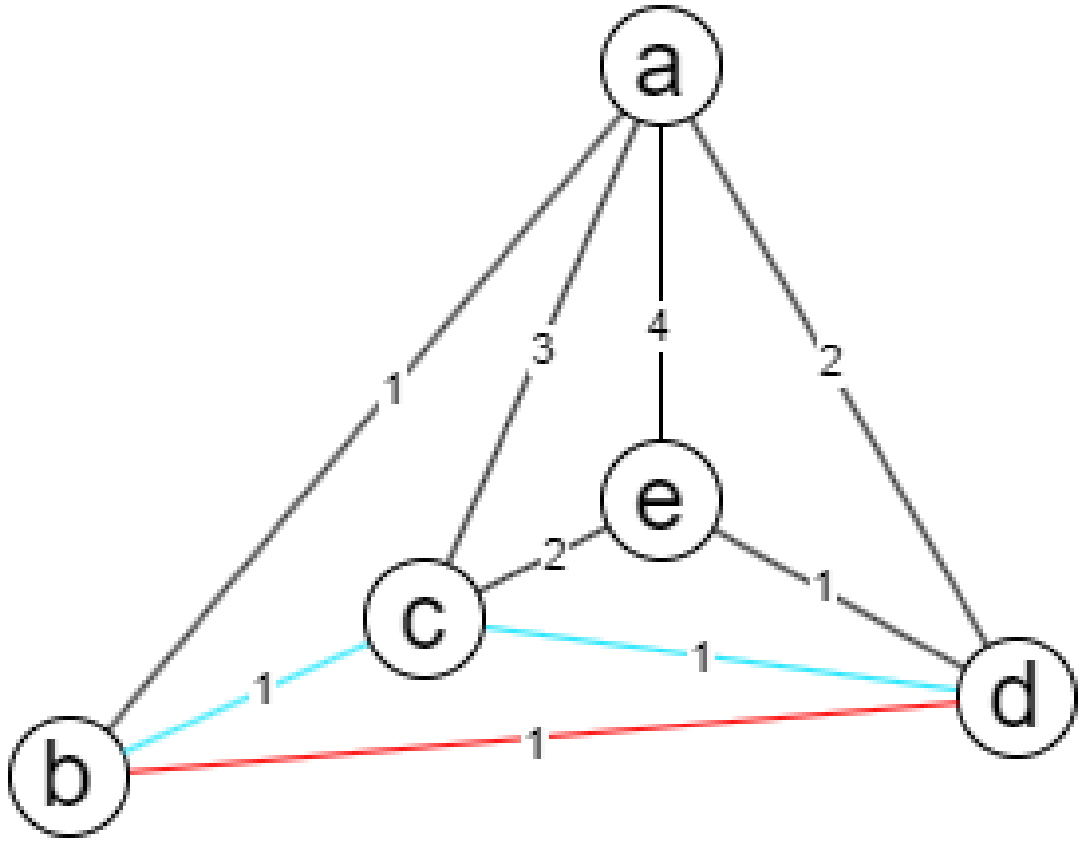
Tillämpning av Kruskals algoritm

- Mål: 5-1=4 bågar
- 2 bågar



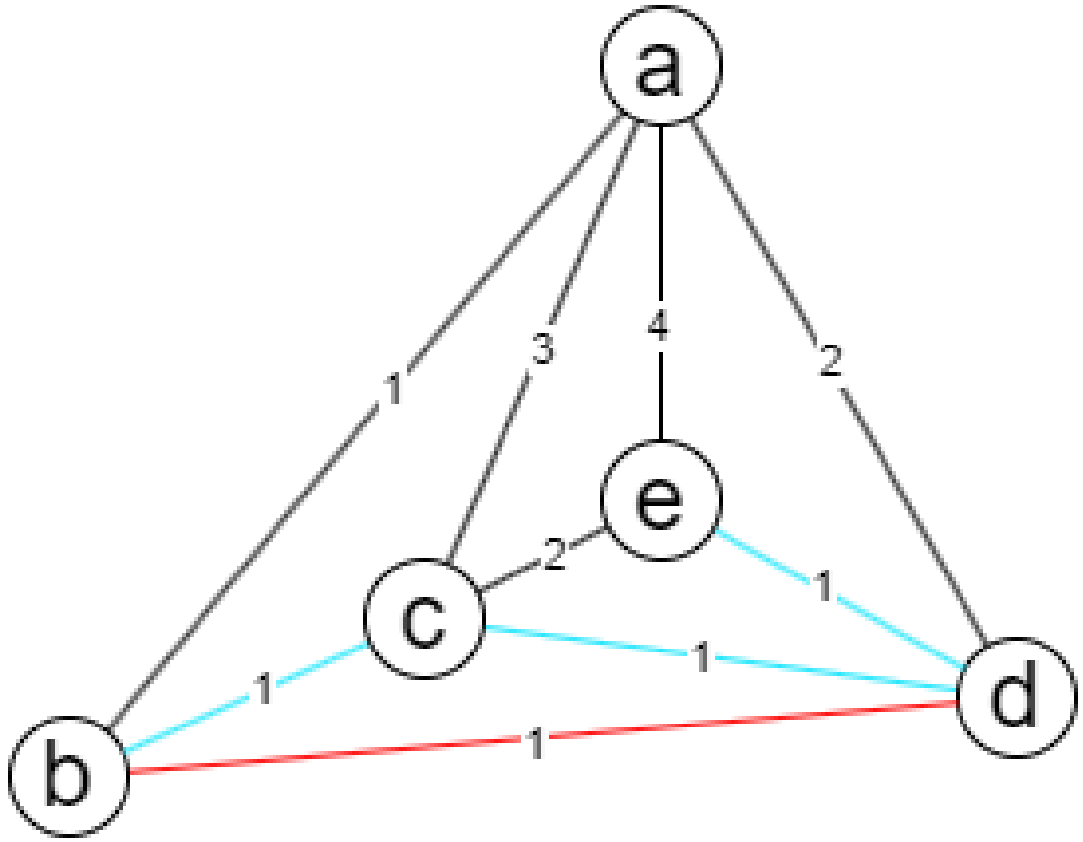
Tillämpning av Kruskals algoritm

- Mål: 5-1=4 bågar
- 2 bågar
- $\{b, d\}$ hade introducerat en cykel



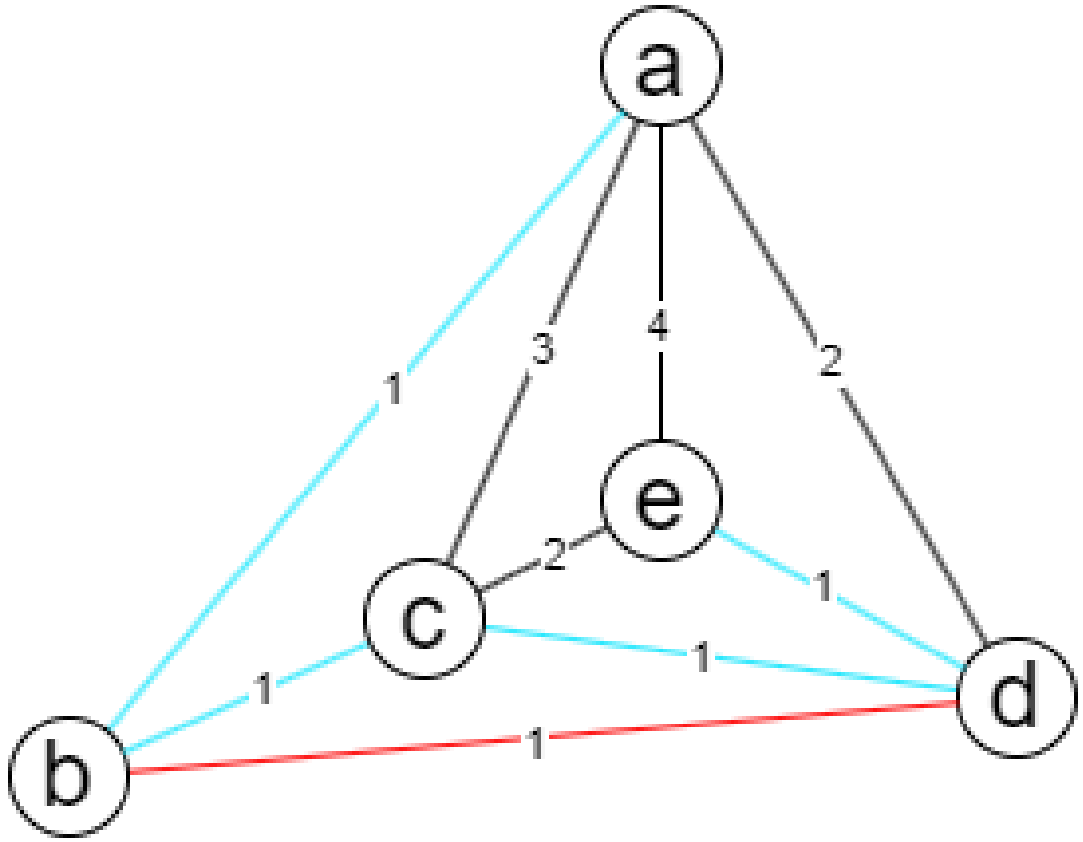
Tillämpning av Kruskals algoritm

- Mål: 5-1=4 bågar
- 3 bågar



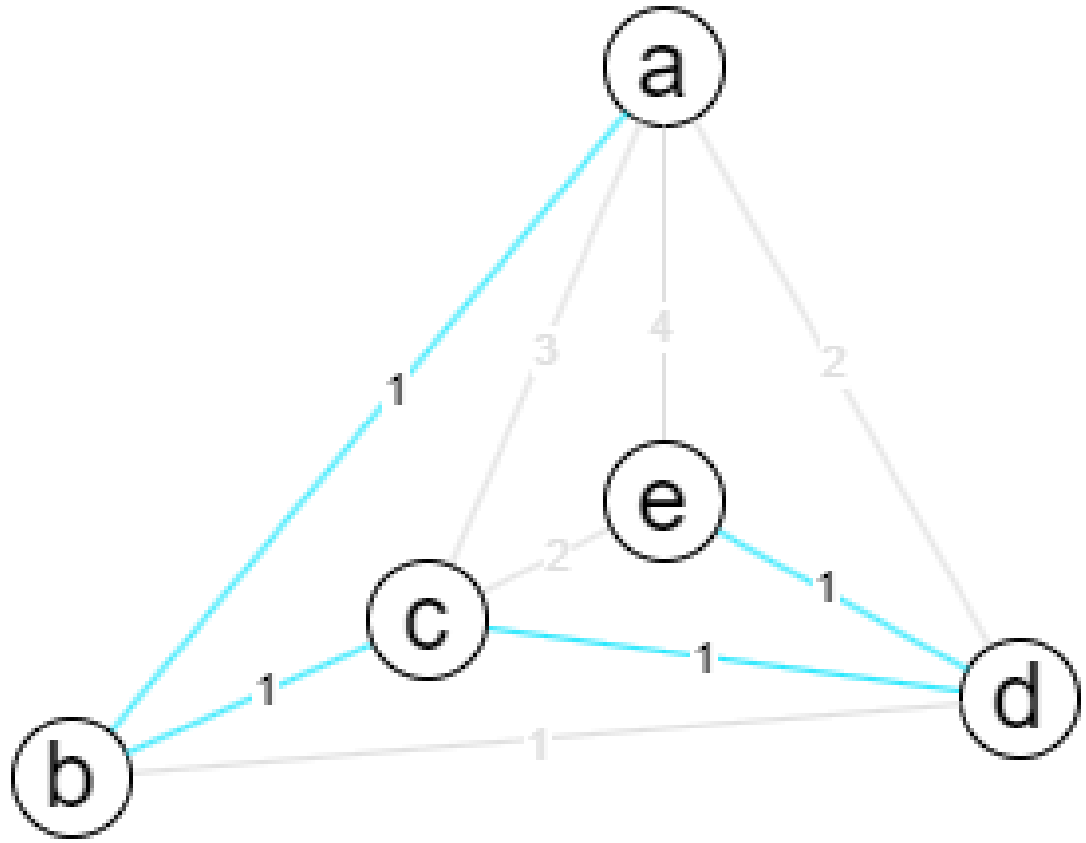
Tillämpning av Kruskals algoritm

- Mål: 5-1=4 bågar
- 4 bågar



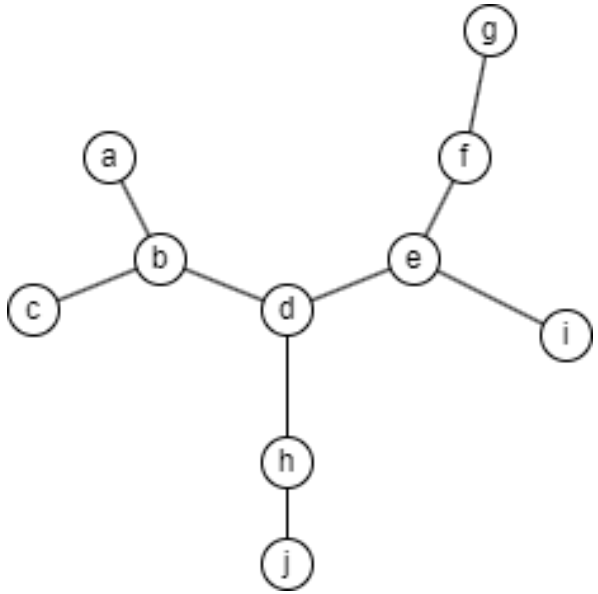
Tillämpning av Kruskals algoritm

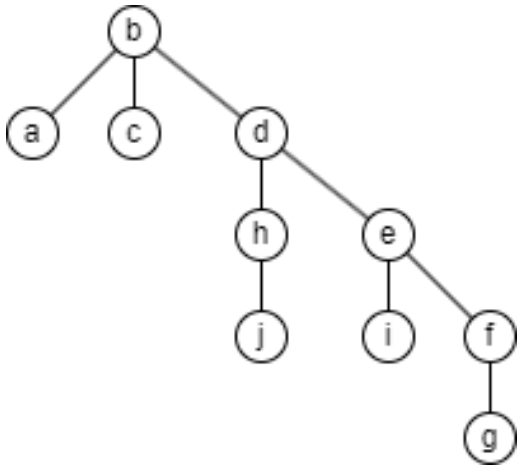
- Mål: 5-1=4 bågar
- 4 bågar

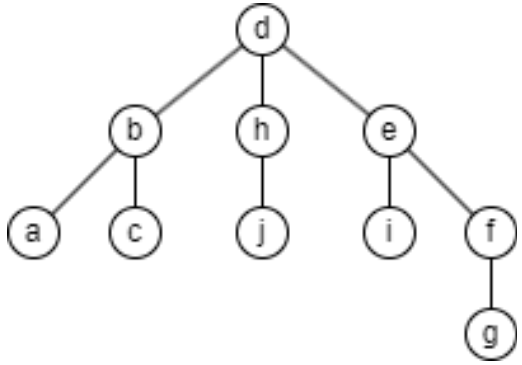


Rotade träd

- Ett träd där en nod definierats som som **rotnod**
- Ett rotat träd får en hierarkisk struktur

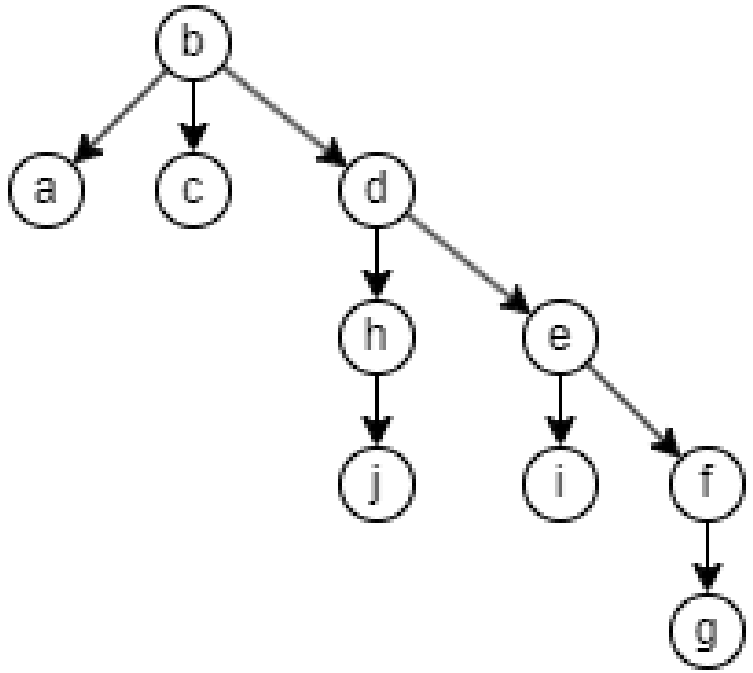






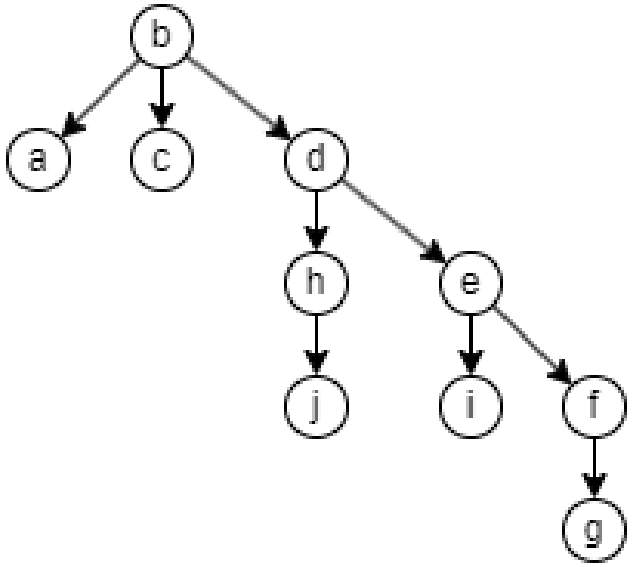
Riktade rotade träd

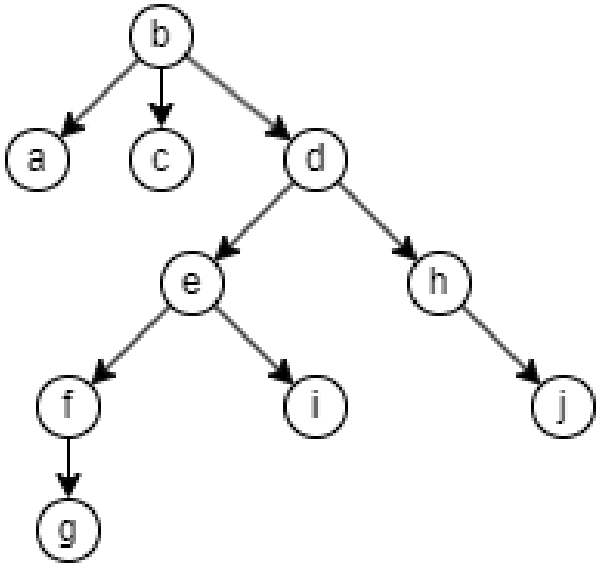
- Vi kan självklart även introducera riktning i ett rotat träd



Rotade träd med ordning

- Vi kan även introducera ordning i ett träd
- Dvs att ordningen mellan barnen i ett träd är viktig (ex. ett släktträd med enbart döttrar skulle kunna ordnas efter födsloordning i varje generation)





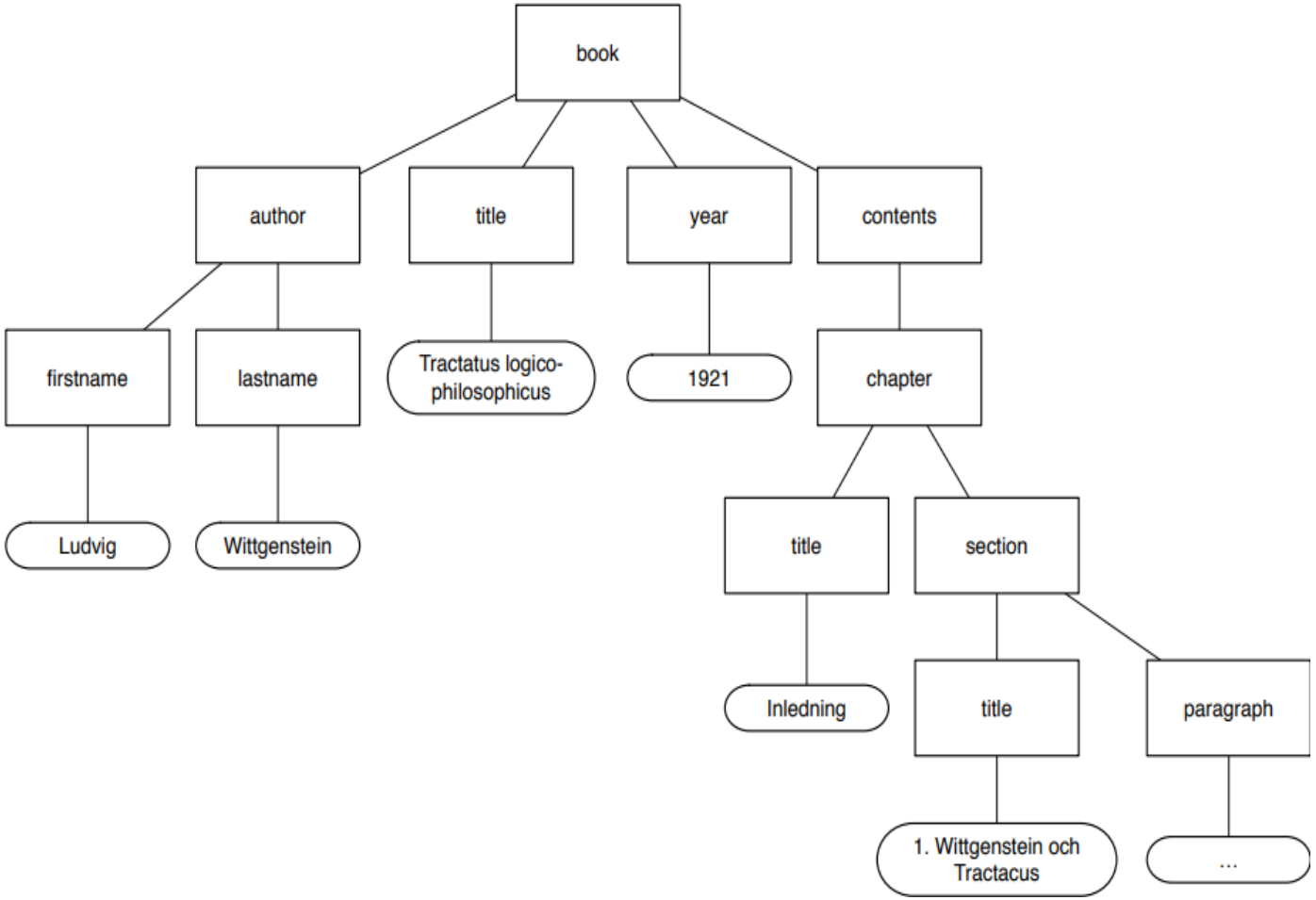
XML - hierarkisk informationsstruktur

- XML är en hierarkisk informationsstruktur
  - Alla element förutom rot-elementet har exakt en förälder
  - Rot-elementet har exakt 0 föräldrar

XML-exempel

```
<book>
    <author>Ludwig Wittgenstein</author>
    <title>Tractatus logico-philosophicus</title>
    <year>1921</year>
    <contents>
        <chapter>
            <title>Inledning</title>
            <section>
                <title>1. Wittgenstein och Tractatus</title>
                <paragraph>
                ...
                </paragraph>
            </section>
        </chapter>
    </contents>
</book>
```

XML-exempel


book
author
title
year
contents
firstname
lastname
Tractatus logico-philosophicus
1921
chapter
Ludvig
Wittgenstein
title
section
Inledning
title
paragraph
1. Wittgenstein och Tractacus
...

Binärt träd

- Speciell typ av träd där varje nod får ha som mest två barn
- Kan vara riktade eller oriktade
- Om det är ordnat: vänster barn, höger barn
- Exempel på tillämpning: Huffmanträd för komprimering (mer på Begreppssem 2)