# MAT1030 – Diskret Matematikk

## Forelesning 24: Grafer og trær

Dag Normann

Matematisk Institutt, Universitetet i Oslo

21. april 2010

(Sist oppdatert: 2010-04-21 12:55)

# Grafteori



## Oppsummering

- Vi har sett på isomorfibegrepet for grafer.
- To grafer er isomorfe hvis alle de viktige egenskapene er de samme.
- Mer presist:

  Det fins en bijeksjon mellom nodene og mellom kantene slik at bildet av en kant går mellom bildet av to noder hvis og bare hvis kanten går mellom nodene.
- Vi definerte stier og kretser
- En sti er en følge av noder og kanter slik at vi går fra node til node via kantene mellom dem.
- En krets er en sti som begynner og slutter samme sted.
- To kretser er like uavhengig av hvor vi starter kretsen som sti, og uavhengig av retningen vi oppgir for stien.
- For en presis definisjon trenger vi å bruke en ekvivalensrelasjon på mengden av stier.



## Oppsummering

- En Eulerkrets er en krets som inneholder hver kant nøyaktig én gang.
- En Eulersti er en sti med samme egenskap.
- En sammenhengende graf har en Eulerkrets hvis graden til alle nodene er et partall.

  En slik graf kalles en Eulergraf.
- En sammenhengende graf har en Eulersti hvis høyst to noder har et oddetall som grad.

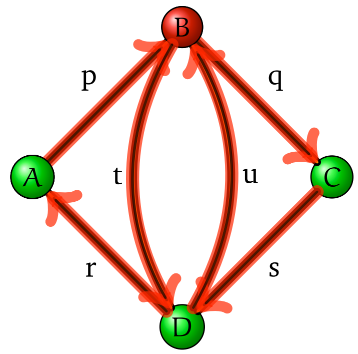
  En graf som har to noder med odde grad er semi-Euler.
- Vi beskrev en pseudokode for å finne en Eulerkrets i en Eulergraf.
- I dag skal vi gi et fullstendig bevis for teoremet om Eulergrafer, men først skal vi repetere pseudokoden:

## Oppsummering

1. Input en Eulergraf $G$ med noder $V$ og kanter $E$
2. $\mathtt{krets} \leftarrow$ en node fra $V$
3. **While** $E \neq \emptyset$ **do**
    - 3.1. $i \leftarrow$ den første noden i $\mathtt{krets}$ med en kant fra $E$ som ligger inntil $i$
    - 3.2. $v \leftarrow i$; $\mathtt{nykrets} \leftarrow i$
    - 3.3. **Repeat**
        - 3.3.1. $e \leftarrow$ en kant fra $E$ som ligger inntil $v$
        - 3.3.2. $v \leftarrow$ noden som er nabo med $v$ via $e$
        - 3.3.3. $\mathtt{nykrets} \leftarrow$ sammensetningen av $\mathtt{nykrets}$ og $e$ og $v$
        - 3.3.4. $E \leftarrow E - \{e\}$

      **until** ingen kant fra $E$ ligger inntil $v$
    - 3.4. $\mathtt{krets} \leftarrow$ sammensetningen av $\mathtt{krets}$ før $i$, $\mathtt{nykrets}$ og $\mathtt{krets}$ etter $i$
4. Output $\mathtt{krets}$






$$E = \{p, q, s, r, t, u\}$$
$$i = AB$$
$$\mathtt{krets} = ApBtDuBqCsDrA$$
$$\mathtt{nykrets} = ApBqCsDrABtDuB$$



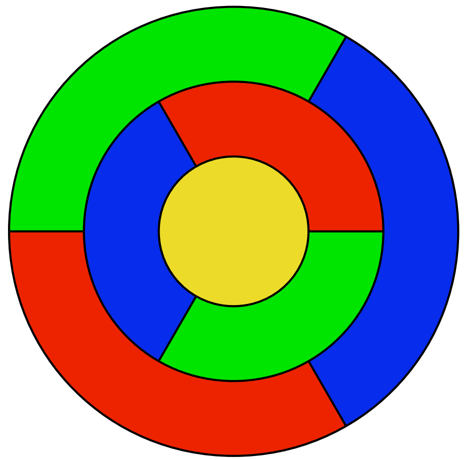
## Digresjon: Firefargeproblemet

- I mange, mange år var følgende et åpent matematisk problem:

  *Anta at vi har et plant kart over landområder (land, fylker, stater o.l.). Er det alltid mulig å trykke kartet ved hjelp av bare fire farger slik at to landområder som grenser opp mot hverandre alltid har forskjellig farge?*
- Hvis vi representerer landene som noder og grensene som kanter, er dette egentlig et grafteoretisk problem.
- Grafteori, som en matematisk tung disiplin, har mye å hente fra forsøkene på å løse dette problemet.
- Måten problemet ble løst på har interesse i seg selv.
- De som løste det, reduserte problemet til et stort antall enkelttilfeller, som deretter ble sjekket av en datamaskin.
- Var det mennesker eller datamaskinen som løste problemet?



## Digresjon: Firefargeproblemet

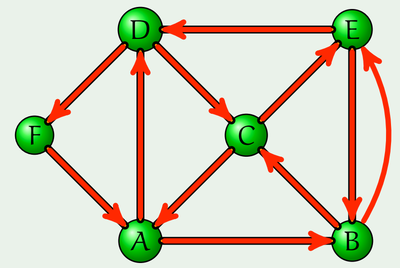


# Eulerstier en gang til

- Vi har sett på en algoritme som fant en Eulerkrets når det var mulig.
- Den kan også brukes til å finne en Eulersti, det vil si en sti som er innom alle kantene nøyaktig én gang, men som kan begynne og slutte på forskjellige steder.
- Disse stedene må da være de to nodene med odde grad.
- Utvider vi grafen med en kant mellom disse to nodene, kan vi lage en Eulerkrets.
- Tar vi bort den nye kanten, får vi en Eulersti.



**Eksempel**






# Eulerstier en gang til

- Det er to strategier for å bevise en setning som Eulers.
- Vi kan bruke et induksjonsbevis.
- Induksjonsbevis gir ofte opphav til algoritmer, og den algoritmen boka presenterer kan sees på som utledet av et induksjonsbevis.
- Vi kan også bevise teoremet *direkte*, uten å argumentere via algoritmen.
- Her følger et direkte bevis for at det er alltid fins en Eulerkrets hvis hver node har grad lik et partall.
- Det er også mulig å trekke en algoritme ut av dette beviset, og algoritmen blir veldig lik den vi så sist.



# Eulerstier en gang til

**Bevis**

La $G$ være en sammenhengende graf hvor gradene til alle nodene er partall. La $S$ være en sti

$$v_0 e_1 v_1 e_2 v_2 \dots e_n v_n$$

av maksimal lengde slik at ingen kant forekommer to ganger. Vi skal vise at $S$ er en Eulerkrets. Vi skal gjøre dette ved å vise følgende tre påstander:

(a) $S$ er en krets.
(b) $S$ inneholder alle nodene i grafen.
(c) $S$ inneholder alle kantene i grafen.

Siden ingen kant forekommer to ganger, må $S$ være en Eulerkrets.

## Eulerstier en gang til

### Bevis (Fortsatt)

(a) Noden $v_0$ må være lik $v_n$. Når vi går ut av den første noden, $v_0$, via kanten $e_1$, så bruker vi opp én av kantene som ligger inntil $v_0$. For hver node vi går inn i og ut av, så bruker vi opp to kanter. Når vi er fremme ved den siste noden i stien, $v_n$, så fins det ingen ubrukt kant som ligger inntil $v_n$. Hadde det vært en slik kant, så ville vi hatt en sti som var lenger enn S, og da hadde ikke S vært maksimal. Siden graden til $v_n$ er et partall, så må vi tidligere i stien ha gått ut av $v_n$. Den eneste muligheten er at $v_n$ er lik $v_0$. Dermed er S en *krets*.



## Eulerstier en gang til

### Bevis (Fortsatt)

(b) S må bestå av alle nodene i grafen. Det er fordi grafen er sammenhengende og S er maksimal.
Hvis en node $v$ ikke hadde vært med, så kunne vi ha laget en sti som var lenger enn S.



## Eulerstier en gang til

### Bevis (Fortsatt)

(c) S inneholder alle kantene fra grafen.
Anta for motsigelse at det fins en kant $e$, som forbinder nodene $u$ og $v$, som ikke er med i S.
Siden S inneholder alle nodene fra grafen, så må $v$ være lik $v_k$ for en passende $k$.
Da kan vi lage en sti som er lenger enn S ved å begynne med $ue$ og fortsette med S:

$$ue\underbrace{v_k e_{k+1} v_{k+1} \dots e_n v_n e_1 v_1 e_2 v_2 \dots e_{k-1} v_{k-1} e_k v_k}_{S}$$



## Hamiltonstier

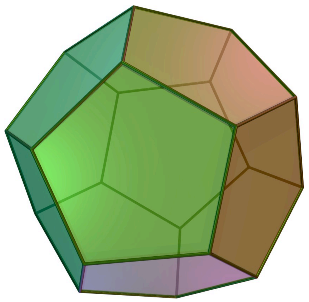
- Vi må også si litt om stier som inneholder alle *nodene* i en graf, uavhengig hvorvidt alle kantene er med eller en kant er med flere ganger.
- "Den handelsreisendes problem" er et slikt problem, hvor man er ute etter den *korteste* stien som går gjennom alle byene i en mengde.

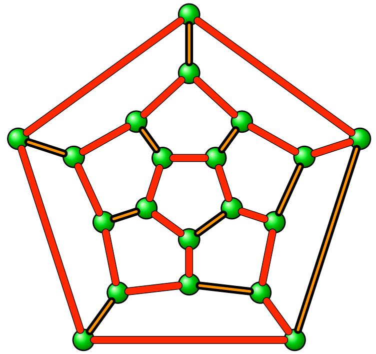
### Definisjon

La G være en sammenhengende graf. En *Hamiltonsti* er en sti som inneholder hver node fra G nøyaktig én gang. En *Hamiltonkrets* er en Hamiltonsti hvor den første og den siste noden sammenfaller. En sammenhengende graf som har en Hamiltonkrets kalles *Hamiltonsk*.

## Hamiltonstier

- Hamiltons puzzle tar utgangspunkt i et dodekaeder (et av de fem Platonske legemene) hvor hvert hjørne er merket med navnet på en by. Spørsmålet han stilte var om det var mulig å reise gjennom alle byene nøyaktig én gang. Vi ser at dette spørsmålet er det samme som om den tilhørende grafen har en Hamiltonsti.







## Hamiltonstier

- Euler studerte også et tilsvarende problem: når det er mulig for en springer å gå over *alle* rutene på sjakkbrett av ulike størrelser.
- Det er ingen som har klart å lage en *effektiv* algoritme for å finne ut om det fins en Hamiltonkrets i en graf.
- Dette er "like vanskelig" som å bestemme om et utsagnslogisk utsagn er en tautologi.

  [Det tilhører klassen av **NP**-komplette problemer.]
- I praksis er det sjeldent at man virkelig trenger å finne en Hamiltonkrets.
- Ofte er det tilstrekkelig å finne en Eulerkrets, eller greit å gå over noder flere ganger.
- Det fins mange spesialtilfeller og heuristikker man kan benytte seg av.



# Kapittel 11: Trær



## Trær

- Et tre er en spesiell type graf.
- Intuitivt er et tre noe som vokser fra en rot

  og så forgrener seg uten noe sted å vokse sammen igjen.
- Vi kan se på et biologisk tre som en graf, ved å la hvert forgreningspunkt være nodene, og delene av en stamme, gren eller kvist mellom to forgreningspunkter være kantene.
- Vi skal gi en presis definisjon av når en graf kan betraktes som et tre.
- Men, hvorfor skal vi lære om trær?

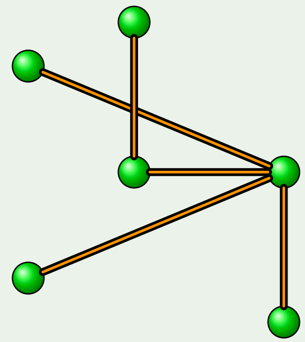
## Trær

- Hvis en graf representerer et nettverk, vil et tre svare til et nettverk hvor det bare fins én sti fra en node til en annen.
- Hvis nettverket består av kabler eller andre medier som formidler informasjon, kan det være hensiktsmessig at signaler bare går langs én vei, slik at systemet ikke forstyrres av at samme informasjon kommer med små tidsforskjeller.
- Dataobjekter som sammensatte algebraiske uttrykk, utsagnslogiske formler eller program har ofte en trestruktur som beskriver hvordan komplekse objekter er bygget opp fra enklere objekter.
- For å undersøke om et utsagn formalisert i matematikken kan bevises eller ikke, kan man prøve å bygge opp et tre av utsagn hvor forgreningen stopper når vi har nådd aksiomene.
- Denne naive idéen danner grunnlaget for enkelte automatiske bevissøkere.

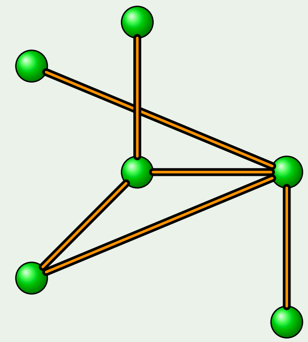


## Trær

**Eksempel**



**Eksempel**





## Trær

**Definisjon**

a) En sykel (engelsk: *cycle*) i en graf er en sti med følgende egenskaper.
   - Stien inneholder minst en kant.
   - Ingen kant forekommer mer enn én gang.
   - Stien er en krets, det vil si, den begynner og slutter i samme node.

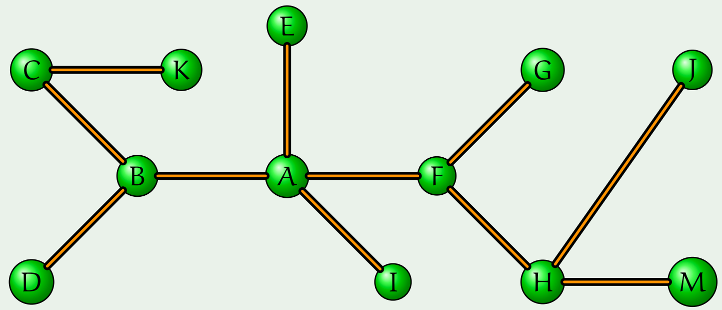
   En sykel med $n$ kanter kalles en $n$-sykel.

b) En graf er et tre hvis grafen er sammenhengende og grafen ikke inneholder noen sykler.



## Trær

**Eksempel**




Vi ser at denne grafen har 12 noder og 11 kanter.

## Trær

### Eksempel

Et tre trenger ikke å ha noen forgreningspunkter:


Her har vi 5 noder og 4 kanter.



## Trær

- I de eksemplene vi har sett på har vi alltid endenoder i et tre, det vil si noder av grad 1.
- Husk at en graf alltid har minst en node.

  Grafen med en node og ingen kanter er et tre. Alle andre trær vil ha endenoder.
- I de eksemplene vi har sett har alle trærne en node mer enn de har kanter.

  Dette er en egenskap som alle endelige trær har.

  Det er ingenting i definisjonen av grafer og trær som sier at de skal være endelige, men vi kommer til å begrense oss til endelige grafer og trær hvis vi ikke sier noe annet. Boka forutsetter også at vi bare arbeider med endelige grafer og trær i dette kurset.



## Trær

### Teorem

a) Hvis et tre har minst en kant, så har treet en node med grad 1 (En slik node kaller vi en endenode eller bladnode).

b) I ethvert tre fins det nøyaktig én node mer enn det fins kanter.



## Trær

### Bevis

a) La

$$v_0 e_1 \cdots e_n v_n$$

være en sti med maksimal lengde hvor ingen kant forekommer to ganger.

Siden grafen er et tre, kan ikke stien være innom samme node to ganger.

Endenodene $v_0$ og $v_n$ må være bladnoder, siden vi ellers ville kunnet gjøre stien lengere.

# Trær

## Bevis (Fortsatt)

b) Vi bruker induksjon på antall noder i treet.

Hvis det bare fins en node, har vi ingen kanter, og påstanden stemmer.

Hvis det fins mer enn en node, kan vi anta at påstanden holder for alle mindre trær.

Tar vi bort en bladnode og den ene kanten som ligger inntil denne noden, får vi et mindre tre.

Siden vi har tatt bort en node og en kant, og ved induksjonsantagelsen da har en node mer enn vi har kanter, må dette være tilfellet i det opprinnelige treet også.

Resonementet illustreres på tavla.



# Vektede grafer

- Hvis en graf representerer et veinett er det av interesse å vite hvor lange de enkelte veistrekningene er.
- Hvis en graf representerer et ledningsnett, kan anleggskostnader og driftskostnader ved de enkelte strekningene være av interesse.
- Hvis nodene i en graf står for land og kantene for grenseoverganger mellom dem, kan tollsatsene eller andre egenskaper ved de forskjellige grenseovergangene bety noe.
- Siden vi har mange eksempler på grafer hvor det er viktige tallstørrelser knyttet til de enkelte kantene, studerer vi vektede grafer som et eget begrep.



# Vektede grafer

## Definisjon

En vektet graf er en graf hvor hver kant har fått en vekt, et positivt reelt tall.

## Merk

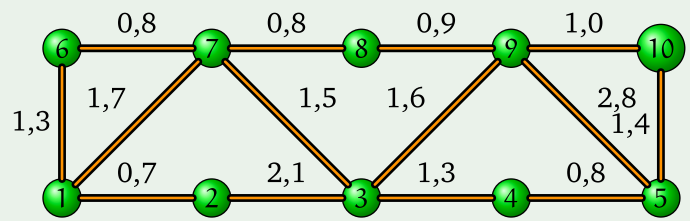
- Formelt sett kan vi definere en vektet graf som et par $(G, f)$ hvor $G$ er en graf og $f$ er en funksjon fra mengden av kanter i $G$ til de positive reelle tallene.
- Vi har altså bruk både for ordnede par og for funksjoner for å gi en skikkelig definisjon.



# Vektede grafer

## Eksempel

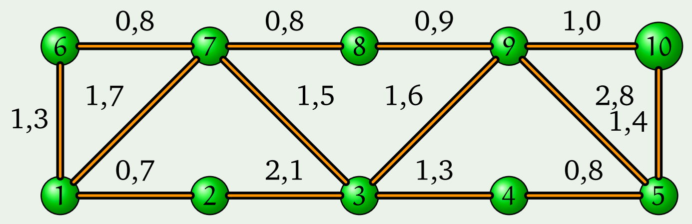
La oss se på et eksempel:




- Det er nå mulig å trekke kabler mellom disse skjematisk tegnede byene, hvor kostnaden f.eks. er målt i antall NOK $10^7$.
- Kan vi fjerne noen av kantene slik at anleggskostnadene blir minst mulig, men vi fortsatt forbinder alle byer med kabler?

# Vektede grafer

## Eksempel (Fortsatt)




- Så lenge grafen inneholder kretser, må det være greit å ta bort en kant i kretsen.
- Vi bør derfor finne det mest kostnadseffektive deltreet som når over alle nodene.
- Vi skal komme tilbake til dette eksemplet når vi har diskutert algoritmen som ligger bak.



# Utspennende trær

## Definisjon

- La G være en sammenhengende graf, og la T være et deltre av G. Det betyr her at T og G har de samme nodene, alle kantene i T er kanter i G, men noen kanter i G kan mangle i T.
- Vi sier at T spenner ut G hvis alle nodene i G ligger inntil en kant i T. (Tegning på tavla.)
- Husk at et tre er en sammenhengende graf, så dette betyr at alle par av forskjellige noder i G kan forbindes med en (og bare en) sti i T.
- Hvis G er en vektet graf, er problemet å finne et tre T som spenner ut G slik at summen av vektene på kantene i T blir minst mulig.



# Utspennende trær

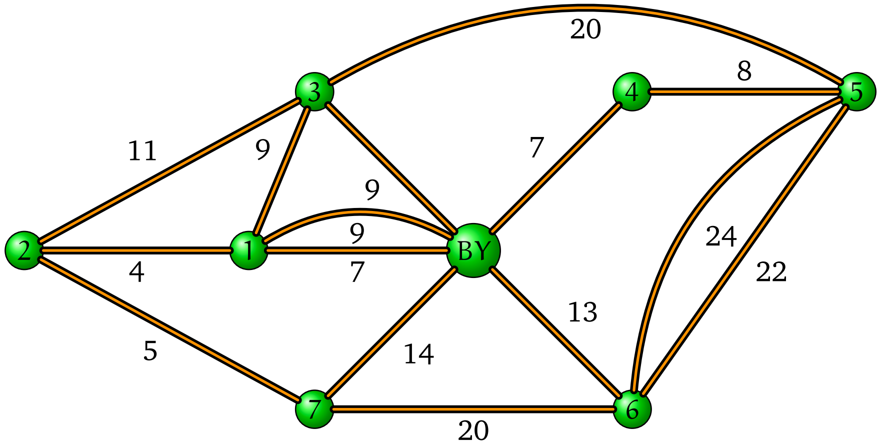
- Det kan finnes mange forskjellige utspennende trær i en graf.
- Hvert slikt tre vil ha en samlet vekt, ved at vi legger sammen vektene på kantene.
- I en situasjon hvor vektene representerer kostnader, og hvor det er teknologisk nødvendig eller tilstrekkelig å erstatte grafen med et utspennende tre, er det av interesse å kunne finne et utspennende tre med minst mulig vekt.
- Det fins effektive algoritmer for å kunne gjøre dette.
- Vi skal se på en slik algoritme: Prims algoritme.



# En kommunegraf

- Vi skal se på et realistisk eksempel på en situasjon som langt på vei kan modelleres som en vektet graf, og hvor det vil være relevant å finne en Eulerkrets eller sti, en Hamiltonkrets og et minimalt utspennende tre for å løse visse samfunnsoppgaver.
- I virkelighetens verden finner man ofte ikke en Eulersti når man trenger en eller en Hamiltonkrets når man trenger en, men som vi skal se, kan man alltid finne minimale utspennende trær.
- Vårt eksempel er en graf som modellerer veinettet mellom de lokale tettstedene i en kommune, og vektingen av kantene er antall kilometer hver enkelt veistrekning er på. Grafen er ikke *enkel*, men bortsett fra det er den som en vektet graf.

## En kommunegraf




- Snøbrøyterne: *Fins det en Eulersti?*
- Postutkjørerne: *Fins det en Hamiltonkrets?*
- Bredbåndutbyggerne: *Fins det et minimalt utspennende tre?*



## En kommunegraf

**Oppgave**

a) Avgjør om det fins en Eulerkrets eller en Eulersti i kommunegrafen, og finn i så fall denne.

b) Er spørsmålet om det fins en Hamiltonkrets det rette spørsmålet?
Kunne postutkjørerne stilt et mer fornuftig grafteoretisk spørsmål?

c) Finn et minimalt utspennende tre (bruker stoff fra resten av forelesningen).
For å få en vektet graf i tråd med definisjonen, kan du ta bort unødige kanter med mye vekt der det fins parallelle kanter.



## Prims algoritme

- Prims algoritme gir en metode for å finne det minimale utspennende treet til en vektet graf.
- I læreboka står det en pseudokode for Prims algoritme.
- Her vil vi beskrive algoritmen litt mer uformelt.
- Det viser seg at hvis man bygger opp et tre ved i hvert skritt å gjøre det som i øyeblikket virker mest fornuftig, så kommer man frem.
- Vi skal trolig ikke gi et korrekthetsbevis for Prims algoritme, men det forventes at man kan praktisere den på eksempler.
- Vi beskriver Prims algoritme litt annerledes enn den er formulert i læreboka, men effekten er den samme, vi får det samme treet bygget opp i den samme rekkefølgen.



## Prims algoritme

- La $G$ være en vektet, sammenhengende graf med noder $V = \{v_1, \ldots, v_n\}$.
- La $T_1$ være treet som består av $v_1$ og ingen kanter.
- Start med node $v_1$ og la $V_1 = \{v_2, \ldots, v_n\}$, altså resten av nodene.
- Finn $v_{i_1} \in V_1$ med minimal avstand til $v_1$ via kant $e_1$.
- Vi får $V_2$ ved å fjerne $v_{i_1}$ fra $V_1$ og vi får $T_2$ ved å legge til $v_{i_1}$ og $e_1$ til $T_1$.
- Deretter fortsetter vi med alltid å velge den ubrukte noden som ligger nærmest, via en kant, til treet bygget opp så langt, og vi bygger ut treet med denne noden og den tilsvarende kanten.
- Siden grafen er sammenhengende, vil vi alltid finne en ny node som er "nabo" til treet bygget opp på et gitt tidspunkt, og da finner vi alltid en ny node som ligger nærmest.
- Vi skal illustrere hvordan denne algoritmen virker på eksemplet vi har gitt på en vektet graf.

# Prims algoritme

## Eksempel (Fortsatt)

- Vi ser på hvordan man ved hjelp av Prims algoritme, skritt for skritt, kan bygge opp et utspennende tre med minimal vekting.
- Vi starter i Node 1.