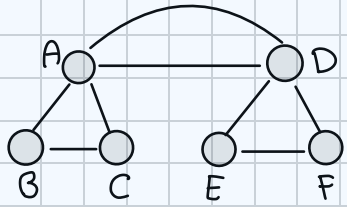
Forelesning 21
GRAFTEORI II

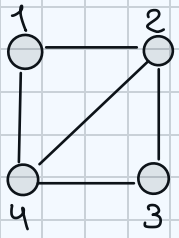
Innhold
- Eulerkretser og -veier, Hierholz algoritme
- Quiz
- Trær
- Rekursiv definisjon av trær
- Spenntrær
- (Vektede grafer)

Sykel: ABCA

Krets: ABCADA (ikke sykel)



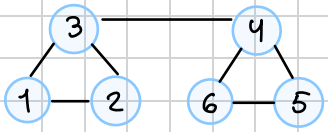
Eulerkrets: ABCADEFDA



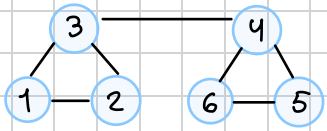
Sykel: 12341
1241

Kretser og sykel

Krets: Lukket vandring uten gjentagende kanter.
Eulervei: Vandring som er innom hver kant nøyaktig en gang.
Eulerkrets: Lukket vandring som er innom hver kant nøyaktig en gang.

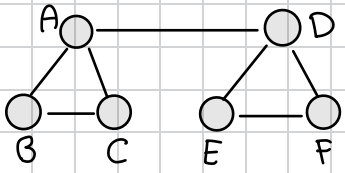


Denne grafen har ingen Eulerkrets. Hvorfor?



Men det er lett å finne en Eulervei (Start i 3).

Når finnes Eulerveier og Eulerkretser?



Eulervei: ABCADEFD

Eulers teorem

(a) En sammenhengende graf har en Eulerkrets hvis og bare hvis alle noder har partallsgrad.

(b) En sammenhengende graf har en Eulervei, men ingen Eulerkrets hvis og bare hvis nøyaktig to noder har oddetallsgrad. Og enhver Eulervei vil da starte i den ene noden med oddetallsgrad, og ende i den andre noden med oddetallsgrad.

Vi har forklart hvorfor det er **nødvendig** med bare noder med partallsgrad for å få en Eulerkrets. Og hvorfor en eventuell Eulervei (som ikke er sykel) må gå mellom to noder med oddetallsgrad, og alle andre noder må ha partallsgrad. Men hvordan kan vi være sikre på at dette er tilstrekkelig? Altså at det faktisk finnes en Eulerkrets eller -vei når betingelsene er oppfylt. Svar: det finnes en algoritme.

Hierholzers algoritme I

Algoritmen finner en Eulerkrets i en sammenhengende graf der alle nodene har partallsgrad.

Ide: Velg tilfeldig startpunkt x_1. Lag en sykel $x_1, x_2, \ldots x_t, x_1$. Hvis ikke alle kanter er brukt, gå tilbake til den siste noden x_j på sykelen som har en ubrukt kant, flett inn en sykel som starter og ender i x_j.

Hvis dette skal gjøres presis og implementeres, brukes en datastruktur som kalles STACK for å holde orden på nodene vi har besøkt, og rekkefølgen.

Dette er en datastruktur der elementer legges på toppen og tas av toppen. Den følger FILO/LIFO-prinsippet. First In, Last Out - det som kom først, går ut sist.

Typiske operasjoner er push (legg på), pop (ta av) og peek/top (se øverste uten å fjerne). Altså som en stabel tallerkener.

Hierholzers algoritme II

Start med en tom liste TUR og en tom STACK. Alle kanter starter "ubrukte".

Velg en startnode A og legg den på STACK.

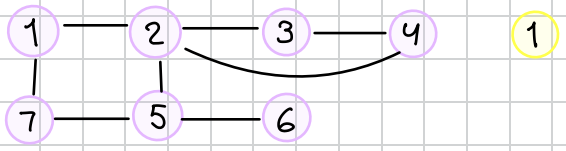
Gjenta følgende, så lenge STACK ikke er tom:

Hvis øverste element X i STACK (peep(STACK)) har en ubrukt kant X-Y, merk denne kanten som "brukt" og legg Y på STACK (push(Y)).

Ellers, fjern X fra STACK (pop(A)), og legg X til tur.

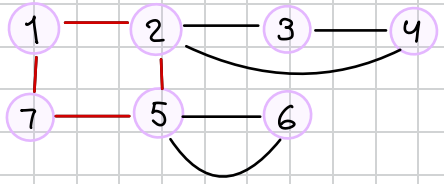
Hierholzers algoritme - eksempel I

Vi skal finne en Eulerkrets i denne grafen. Siden alle noder har partallsgrad, vet vi at en slik finnes. Vi starter med å tilfeldig velge node 1, og legger det på Stacken (i gult).



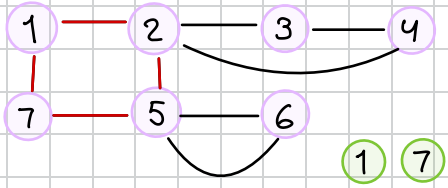
Hierholzers algoritme - eksempel II

I de første stegene velger vi (i rekkefølge) kantene 1-2-5-7-1, merker de som brukte, og legger på Stacken.



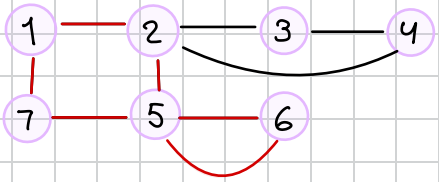
Hierholzers algoritme - eksempel III

Nå overfører vi først 1, så 7 fra Stacken til turen (i grønt), siden de ikke har noen ubrukte kanter:



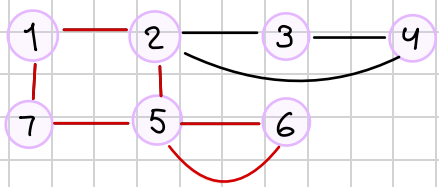
Hierholzers algoritme - eksempel IV

Nå ligger 5 øverst på Stacken, vi fyller inn kretsen 5-6-5, når vi velger den ene kanten 5-6 blir 6 lagt på Stacken, når vi velger den andre kanten blir 5 lagt på Stacken.



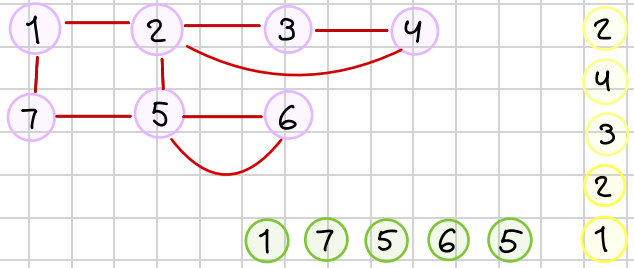
Hierholzers algoritme - eksempel V

Så overfører vi først 5, så 6, så 5 igjen fra Stacken til turen (i grønt), siden de ikke har noen ubrukte kanter:



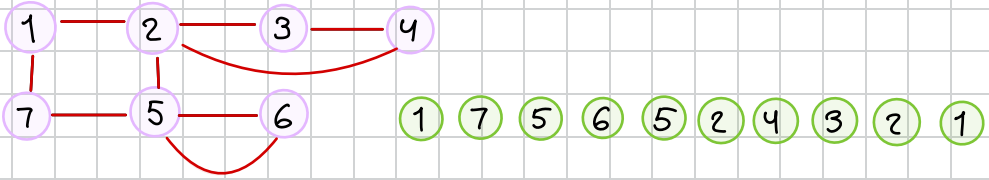
Hierholzers algoritme - eksempel VI

Så starter vi i 2, som ligger øverst i Stacken, og velger kantene 2-3-4-2, mens 3,4 og 2 (i den rekkefølgen) legges til på Stacken.



Hierholzers algoritme - eksempel VI

Nå er alle kanter brukte, så vi tømmer Stacken (alltid fra toppen)



Turen (i grønt) er en Eulerkrets (kan leses fra venstre eller fra høyre).

Quiz

Hvis to grafer G og H har like mange noder og like mange kanter, så må de være isomorfe? Usant

Hvis to grafer G og H er isomorfe, så må de ha samme antall kanter
Sant

Enhver graf G har samme antall noder som komplementet G
Sant

Enhver graf G har samme antall kanter som komplementet G
Usant

La G være en graf med 5 noder og 7 kanter. Hvor mange kanter har komplementet G? 3

En graf G og komplementet G må ha samme antall sammenhengskomponenter?
Usant

Den komplette grafen K_7 med 7 noder har en Eulerkrets
Sant

I et selskap med 7 personer, er det mulig at alle kjenner nøyaktig tre andre gjester?
Nei

Trær

Husk at en sykel er en lukket sti (altså en lukket vandring uten gjentagende noder).

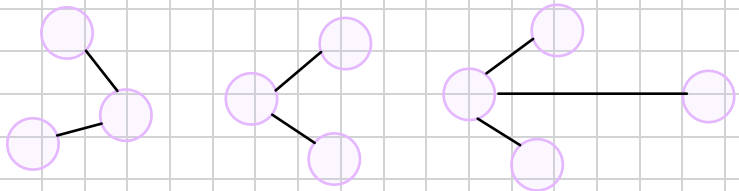
Vi sier at en graf er sykelfri om den ikke inneholder noen sykler.

Definisjon

1. En graf er et tre dersom den er sammenhengende og sykelfri.
2. En graf er en skog dersom den er sykelfri.

Eksempel

Dette er en skog som består av trær.

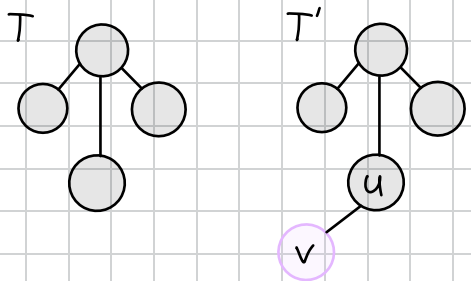


Rekursiv definisjon av trær - ide

Hvis $T=(V,E)$ er et tre og vi lager T' ved å legge til en ny node v og en kant $\{u,v\}$ for en node $u \in V$, så er T' også et tre.

T' er sammenhengende: Den nye noden v er forbundet med T ved den nye kanten $\{u,v\}$, og T er sammenhengende.

T' kan ikke ha sykler: T har ikke sykler, så hvis T' skulle ha en sykel, måtte den passere gjennom den nye noden v. Men v har grad 1, så det kan ikke være noen sykler som inneholder v.



Rekursiv definisjon av trær - formell definisjon

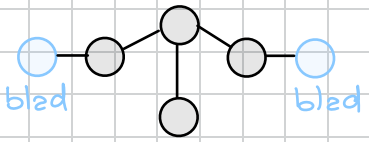
Basis: En graf med en node og ingen kanter er et tre.
Induktivt steg: Hvis $T=(V,E)$ er et tre og $u \in V$, så er grafen T' som fås ved å legge til en ny node v og en kant $\{u,v\}$ også et tre.

Rekursiv definisjon av trær - ekvivalens I

Vi har definert trær på to måter: sammenhengende grafer uten sykler, og den rekursive definisjonen. Vi må forsikre oss om at de to definisjonene er ekvivalente.

Vi må overbevise oss om at alle sammenhengende grafer uten sykler, også er konstruerbare via den rekursive definisjonen.

Vi observerer først at alle sammenhengende grafer, med minst to noder og uten sykler, har minst et blad (en node av grad 1).



Endepunktene til stier av maksimal lengde må være blader.

Rekursiv definisjon av trær - ekvivalens II

La $T=(V,E)$ være en sammenhengende graf uten sykler. Hvis $|V|=1$ er vi i basistilfellet for den rekursive definisjonen. Vi gjør nå et induksjonsargument med hensyn på $|V|$.

Anta at alle sammenhengende grafer med $<k$ noder kan konstrueres rekursivt, og anta nå at T har k noder.

T må ha et blad v. Anta $\{u,v\}$ er den eneste kanten som er koblet til v.

La T^* være grafen vi får ved å fjerne v og $\{u,v\}$ fra T. Da er T^* sammenhengende og uten sykler.

Antall noder T^* er $<k$, og T^* kan derfor konstrueres rekursivt. Hvis vi nå legger til v og $\{u,v\}$ igjen, har vi konstruert også T rekursivt.

Noder = Kanter + 1

Vi kan bruke den rekursive definisjonen til å vise den fine sammenhengen:

$$|V| = |E| + 1$$

for ethvert tre $T=(V,E)$.

Dette holder opplagt i basissteget (en node, null kanter). Og i det induktive steget, legger vi til en kant og en node.

Undergraf

G' er en undergraf av G, dersom vi kan konstruere G' ved å

- → fjerne kanter fra G
- → fjerne noder fra G, slik at hvis vi fjerner en node u, fjerner vi også alle kantene som er koblet til u.

G' er en utspennende undergraf av G, dersom G' og G har de samme nodene (altså vi fjerner bare kanter).

Fakta om trær

- Hvis vi har en sammenhengende graf med sykler, kan vi suksessivt fjerne kant fra sykel, til vi står igjen med et tre.
- Alle trær har derfor en utspennende undergraf som er et tre, dette kalles et spenntre.
- For enhver sammenhengende graf $G=(V,E)$ har vi $|E| \geq |V|-1$, altså
$$|V| \leq |E|+1$$
- For en sammenhengende graf $G=(V,E)$ har $|V|=|E|+1$ hvis og bare hvis grafen er et tre.