Föreläsning 8: Träd och heltalspartitioner · 1MA020

Vilhelm Agdur[1]

[1] vilhelm.agdur@math.uu.se

15 februari 2023

Vi introducerar grafer och träd, och bevisar att antalet rotade ordnade binära oetiketterade träd också räknas av Catalantalen.

Sedan introducerar vi heltalspartitioner, och härleder en genererande funktion för dessa, som vi använder för några exempel.

Grafer och träd

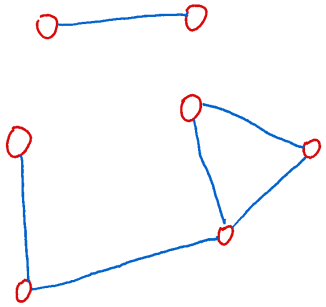
Vårt första ämne i denna föreläsning är grafer och träd, som kommer dyka upp igen och igen också i senare föreläsningar – det är ju till och med den preliminära titeln på vår sista föreläsning. Vi börjar med att ge en samling definitioner av vad vi menar med dessa ord, och sedan börjar vi räkna hur många av olika typer av graf det finns i olika klasser.

Definition 1. En *graf* består av en mängd V av *noder* och en mängd $E \subseteq \binom{V}{2}$ av kanter.[2] Om det finns en kant $\{u, v\}$ säger vi att u och v är *grannar*. En graf är *etiketterad* om noderna är särskiljbara, annars är den oetiketterad. [3] Vi säger att en graf är *sammanhängande* om det går att nå varje nod från varje annan nod genom att vandra längs kanterna. Ett sätt att vandra från en nod tillbaka till sig själv kallar vi för en *cykel*.

[2] Med notationen $\binom{A}{k}$ där A är en mängd och k ett heltal menar vi *mängden* av delmängder av storlek k till n. Alltså har vi att

$$\left|\binom{[n]}{k}\right| = \binom{n}{k}.$$

[3] Det här är precis samma koncept som med våra lådor som var särskiljbara eller inte. Antingen har noderna namn, så vi kan prata om nod nummer tre, eller så kan vi bara se vilka andra noder de har kanter till.



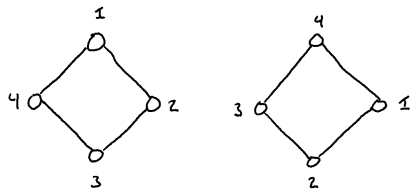
Figur 1: Ett exempel på en graf. Den är inte sammanhängande, eftersom de övre två noderna inte kan nås från de undre fem. Triangeln utgör en cykel, som är grafens enda.

Exempel 2. Det finns $2^{\binom{n}{2}}$ stycken etiketterade grafer på n noder, eftersom det finns $\binom{n}{2}$ möjliga kanter, och vi får en graf per val av vilka kanter som skall vara med.

Problemet med att räkna antalet oetiketterade grafer på n noder är betydligt mer komplicerat. Den första idén man hade haft är kanske att det borde vara

$$\frac{2^{\binom{n}{2}}}{n!}$$

eftersom det borde finnas $n!$ olika sätt att sätta dit etiketterna. Problemet är att vissa grafer har symmetrier som gör att till synes olika sätt att skriva dit etiketter i själva verket ger samma etiketterade graf.



Figur 2: Två till synes olika etiketteringar av samma graf, som i själva verket är samma etikettering på grund av grafens rotationssymmetri.

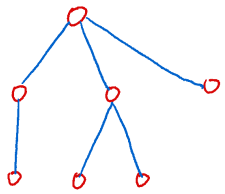
Som tur är visar det sig att nästan alla grafer inte har någon symmetri alls, så svaret är *nästan* $\frac{2^{\binom{n}{2}}}{n!}$.[4]

[4] Det här påståendet låter kanske löst i kanten, men det är faktiskt helt rigoröst. I alla fall om man ersätter "nästan" med att skriva att antalet är

$$(1+o(1))\frac{2^{\binom{n}{2}}}{n!}.$$

Definition 3. Ett *träd* är en sammanhängande graf utan cykler. Ett *rotat* träd är ett träd med en specifik nod utpekad som dess rot.[5] I ett rotat träd har varje nod utom roten själv en granne som är närmre roten än sig[6], vilken vi kallar dess *förälder*. Alla dess andra grannar kallar vi dess *barn*. En nod utan barn kallar vi för ett *löv*, och en nod som inte är ett löv kallar vi för *intern*.

Ifall det spelar roll i vilken ordning vi ritat noderna kallar vi trädet *ordnat*, se figur 4.



Figur 3: Ett träd med sju noder och sex kanter.

[5] Så om trädet är oetiketterat kan vi alltså se vilken nod som är roten, men resten av noderna kan vi inte se skillnad på, bara vilka som hänger ihop med vilka med kanter.

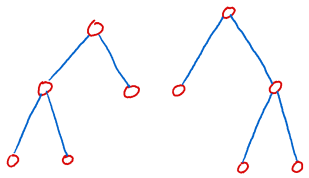
[6] Eller är roten.

Definition 4. Ett träd i vilket alla noder antingen har två eller noll barn kallas för ett *binärt* träd.

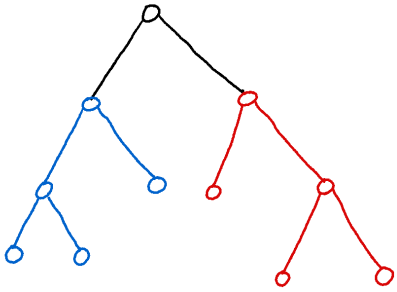
Låt oss nu återse en gammal vän, Catalantalen.

Proposition 5. *Antalet rotade ordnade binära oetiketterade träd med n stycken interna noder ges av Catalantalen.*

Bevis. Vi kan dela upp ett sådant träd i två mindre träd genom att helt enkelt ta bort roten, och låta dess två barn vara rötter i två mindre träd.



Figur 4: Två träd som är olika varandra som ordnade träd, men samma träd som oordnade träd.



Figur 5: Ett rotat ordnat binärt oetiketterat träd, med uppdelningen av det i två mindre träd av samma typ, ett rött och ett blått.

Alltså gäller det, om t_n betecknar antalet sådana träd, att

$$t_{n+1} = \sum_{k=0}^{n} t_k t_{n-k},$$

eftersom vi kan skapa oss ett sådant träd med $n+1$ noder genom att först rita roten, och sedan fästa ett träd med k interna noder till vänster och ett med $n-k$ interna noder till höger. Eftersom roten själv är intern har vi då $k+n-k+1 = n+1$ interna noder. □

Cayleys formel

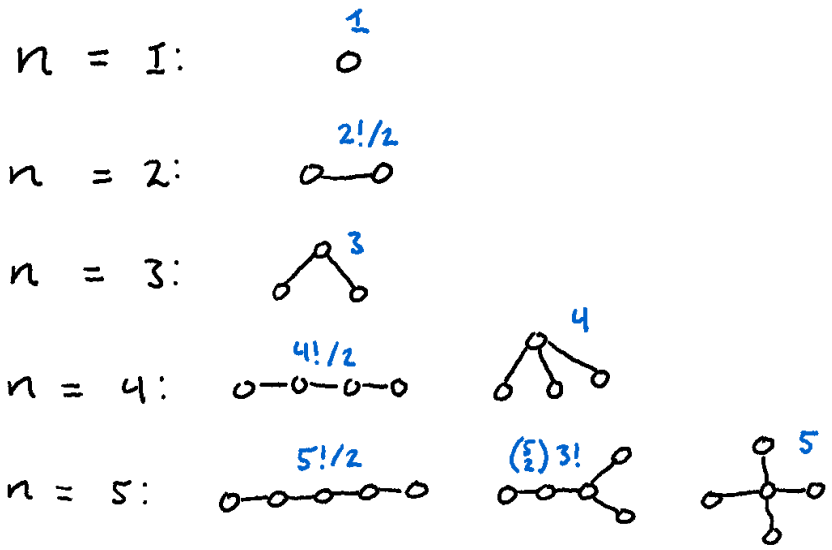
Vi har alltså lyckats räkna en väldigt specifik sorts träd. Kan vi räkna träd mer generellt?

Teorem 6. *Det finns*

$$n^{n-2}$$

stycken etiketterade träd med n noder.

För att förstå detta resultat, låt oss börja med att räkna de första små fallen. Vi kollar på *oetiketterade* träd, och räknar hur många sätt vi kan sätta etiketter på dem.



Figur 6: Oetiketterade grafer med n noder, för $n = 1, \ldots, 5$, med antalet sätt att sätta etiketter på varje i blått.

Sätten vi får dessa antal är, per värde på n:

1. Att det bara finns ett sätt att skriva en etta på den enda noden är uppenbart.
2. Vi kan välja vilken permutation som helst av $[2]$ att skriva på noderna, men grafen har en speglingssymmetri, så att skriva etiketterna i motsatt ordning ger samma träd. Alltså $\frac{2!}{2}$.
3. Vi kan se vilken nod som är den mellersta, men vi kan inte se skillnad på de två yttre. Alltså är det enda val vi kan göra det av vilket tal vi skriver på den mellersta, vilket vi kan välja på tre sätt.

4. För den första av våra två oetiketterade grafer kan vi skriva vilken permutation av $[4]$ vi vill, men återigen har vi en speglingssymmetri, så att skriva den baklänges ger oss samma etikettering. Alltså $\frac{4!}{2}$.

   För den andra är vi i samma situation som vi var i för $n = 3$ – vi kan se vilken nod det är som har mer än en granne, men vi kan inte se skillnad på de andra. Alltså är det enda valet vi har vilken etikett just den särskilda noden får, vilket vi kan göra på 4 sätt.

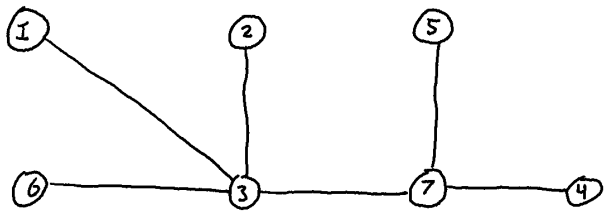
5. Vi får $\frac{5!}{2}$ av samma speglingssymmetri-skäl som innan, och för den tredje av våra grafer får vi 5 eftersom vi åter har en särskild nod och resten kan vi inte se skillnad på.

   För den mellersta av våra tre oetiketterade träd kan vi se skillnad på de tre noderna i svansen till vänster, men de två som sticker ut åt höger kan vi inte se skillnad på. Så för att etikettera denna väljer vi två etiketter för de talen, vilket vi kan göra på $\binom{5}{2}$ sätt, och sedan är varje permutation av de återstående tre etiketterna faktiskt en distinkt etikettering, så vi kan välja 3! sätt att fullfölja vår etikettering. Så vi har totalt $\binom{5}{2}3!$ sätt att göra detta på.

Så vi ser i alla fall att vår formel gäller för n upp till fem. Vi väljer att ge två bevis för denna sats. Det första är av Prüfer, och ger en bijektion mellan etiketterade träd och en enklare mängd.

Prüfers bevis av Cayleys formel (1918). Vi vill visa på en bijektion mellan mängden av etiketterade träd på n noder och mängden av ord av längd $n-2$ ur alfabetet $[n]$. Att den senare mängden har rätt antal element vet vi sedan innan, så om vi kan hitta en bijektion är vi klara.

Vi börjar med att berätta hur vi skapar vårt ord givet ett etiketterat träd.[7] Vi letar upp det löv[8] som har lägst etikett, skriver dess etikett som första bokstav i vårt ord, och tar sedan bort noden ur trädet.



Figur 7: Ett etiketterat träd med Prüferkod 33773.

Vi upprepar denna process – letar upp det löv i det resulterande trädet som har lägst etikett (och notera att vi, när vi tar bort ett löv, ibland kommer göra en nod som innan var intern till ett löv), skriver den etiketten på slutet av ordet, och tar bort lövet. Processen fortsätter tills vi bara har två noder kvar.

[7] Detta ord kallas för trädets *Prüferkod*.

[8] Strikt sett har vi hittills bara definierat *löv* för rotade träd – och de träd vi studerar här har ju ingen rot-nod. Med "löv" menar vi här "nod med bara en granne".

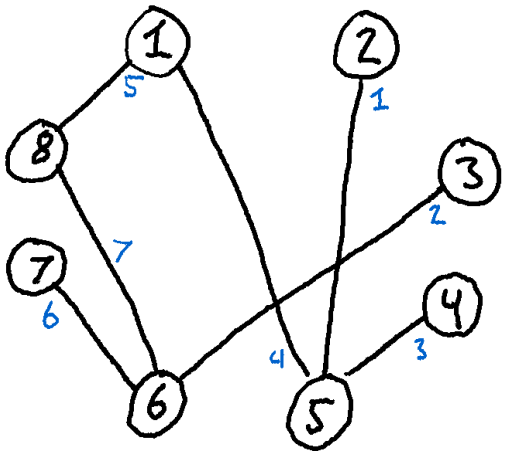
För att gå från en Prüferkod till en etiketterad graf använder vi följande algoritm, som konstruerar ett etiketterat träd givet en Prüferkod $a_1 a_2 \dots a_{n-2}$.

Algorithm 1: Konstruktion av träd från Prüferkoder

Låt G vara en graf med n noder, med etiketter $1, 2, \dots, n$, men utan kanter
$L_1 \leftarrow (1, 2, \dots, n)$
$A_1 \leftarrow (a_1, a_2, \dots, a_{n-2})$
for $t = 1, 2, \dots$ **do**
  if $|L_t| = 2$ **then**
    Rita en kant i G mellan de två elementen i L
    stop
  else
    Låt l vara det minsta elementet i L_t som inte är i A_t
    Låt a vara det första elementet i A_t
    Rita en kant i G mellan l och a
    $L_{t+1} \leftarrow L_t \setminus \{l\}$
    $A_{t+1} \leftarrow (A_t(2), A_t(3), \dots, A_t(|A_t|))$
  end if
end for
return G

Det tar en stund att förstå vad den här algoritmen faktiskt gör[9], men efter en stunds kontemplation ser man att vad den gör är att den lägger till kanterna i precis den ordning som de försvann när vi skapade Prüferkoden.

[9] En fördel med föreläsningar över text är att man faktiskt kan genomföra algoritmen på ett konkret exempel, för att illustrera den, men en text måste så klart vara statisk. Sitter du hemma och läser föreslår jag att du prövar att göra algoritmen för hand på någon eller några Prüferkoder, för att få en känsla för vad som pågår.



Figur 8: Ett träd skapat från Prüferkoden 565186. I blått har vi markerat i vilket steg i algoritmen varje kant lades till – lägg märke till att detta är precis ordningen i vilken de *tas bort* om vi skapar Prüferkoden för detta trädet.

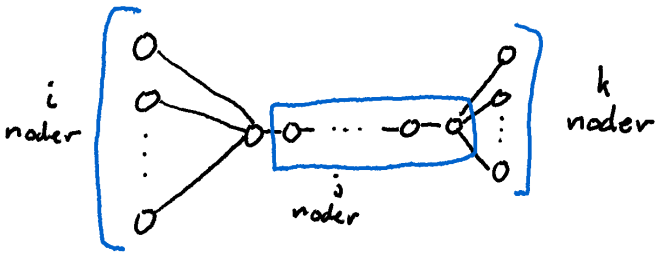
Den första kanten att ritas kommer gå mellan det första lövet att tas bort och dess granne, den andra går mellan andra lövet att tas bort och dess granne, och så vidare – ända fram tills den sista kanten vi lägger till, som går mellan de två kvarvarande noderna när vi

byggde Prüferkoden. Alltså kommer denna algoritmen precis att rekonstruera grafen vi började med, vilket bevisar att vi faktiskt har en bijektion, och satsen följer. □

Övningar

Övning 1. Bevisa att ett träd alltid har $|E| = |V| - 1$.

Övning 2. Överväg följande skiss av ett oetiketterat träd med $1 + i + j + k$ noder:



Figur 9: Skiss av ett oetiketterat träd.

Hur många olika sätt finns det att sätta etiketter på detta träd?[10] Ge en formel som gäller för alla $i, j, k = 0, 1, 2, \ldots$.[11]

Övning 3. Rita det etiketterade trädet som har Prüferkod 1273262.

Övning 4. *En övning för dig som kan programmera.*[12] Implementera vår algoritm för att omvandla en Prüferkod till ett etiketterat träd i faktisk kod. Koden skall ge ett lämpligt grafobjekt som output, och en bild av grafen.[13]

Pröva din kod på några slumpmässigt valda Prüferkoder. Hur ser ett träd vanligtvis ut?

[10] Vi resonerade om detta för några små träd precis efter att vi introducerade Cayleys formel, men här har vi alltså ett mer generellt fall.

[11] Finns det några specialfall för särskilda kombinationer av värden på i, j, och k?

[12] För inlämningsuppgiften i kursen är denna uppgift frivillig om ni inte har någon i er grupp som kan programmering. Om ni genomför den, vänligen använd inte Matlab, eftersom jag inte kan köra sådan kod. Mathematica är okej.

[13] Skriver du i R, C/C++, eller Python kan jag föreslå *igraph*-paketet för graf-datatypen och att rita dem. Mathematica har inbyggd funktionalitet för detta, så klart.