Föreläsning 8: Grafer och träd · 1MA020

Vilhelm Agdur[1]

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

16 februari 2023

Vi introducerar grafer och träd, och bevisar att antalet rotade ordnade binära oetiketterade träd också räknas av Catalantalen. Vi bevisar också att samlingen av alla oetiketterade ordnade träd räknas av Catalantalen, fast på ett lite annat sätt.

Efter det bevisar vi Cayleys formel, som säger att det finns n^{n-2} etiketterade träd på n noder, på två olika sätt.

Grafer och träd

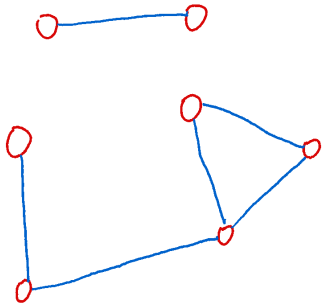
Vårt första ämne i denna föreläsning är grafer och träd, som kommer dyka upp igen och igen också i senare föreläsningar – det är ju till och med den preliminära titeln på vår sista föreläsning. Vi börjar med att ge en samling definitioner av vad vi menar med dessa ord, och sedan börjar vi räkna hur många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



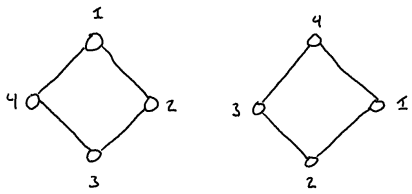
Figur 1: Ett exempel på en graf. Den är inte sammanhängande, eftersom de övre två noderna inte kan nås från de undre fem. Triangeln utgör en cykel, som är grafens enda.

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.

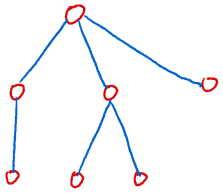
Som tur är visar det sig att nästan alla grafer inte har någon symmetri alls, så svaret är *nästan* $\frac{2^{\binom{n}{2}}}{n!}$.[4]

[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.

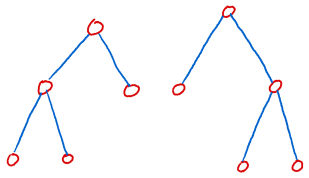


Figur 3: Ett träd med sju noder och sex kanter.

[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.



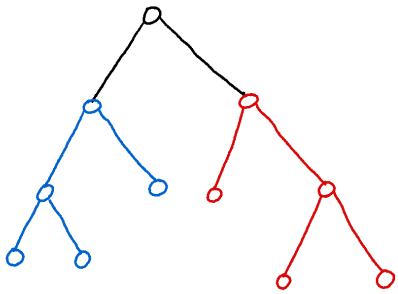
Figur 4: Två träd som är olika varandra som ordnade träd, men samma träd som oordnade träd.

Ännu fler saker som räknas av Catalantalen

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 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. □

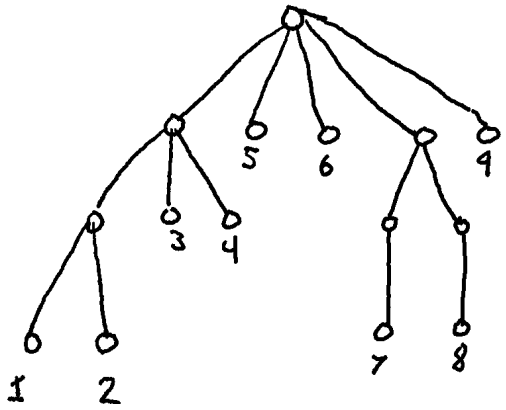
I själva verket räknas också godtyckliga oetiketterade ordnade träd av Catalantalen, utan något krav på att de skall vara binära.

Proposition 6. *Antalet rotade ordnade oetiketterade träd på $n+1$ noder ges av Catalantalen.*[7]

[7] Notera att vi här räknar *alla* noder, inte bara de interna, som vi gjorde för de binära träden.

Bevis. Vi bevisar detta genom att uppvisa en bijektion med sätt att skriva n matchande par av parenteser.

Så, givet ett rotat ordnat oetiketterat träd på $n+1$ noder, numrera dess löv från höger till vänster.



Figur 6: Ett rotat ordnat oetiketterat träd, med dess löv numrerade från vänster till höger.

I vår algoritm för att omvandla detta träd till en parentetisering blir uttrycket, innan vi ersatt talen med $()$,

$$((12)34)56((7)(8))9.$$

Skriv sedan upp talen $12\ldots k$ på rad, och skriv i parenteser på följande vis: För varje intern nod, förutom roten, skriv ett par parenteser runt alla de tal som motsvarar löv under noden. Ersätt sedan varje av talen med ett tomt par av parenteser, $()$. Detta ger en parentetisering med precis n par av parenteser.

För att återhämta ett träd från en parentetisering, börja med att rita en nod för varje $()$. Sedan, för varje grupp av noder som ligger i samma par av parenteser, rita en gemensam förälder för dem. Till slut, rita dit roten, och koppla den till varje par av parenteser som inte har någon parentes runt sig. □

Cayleys formel

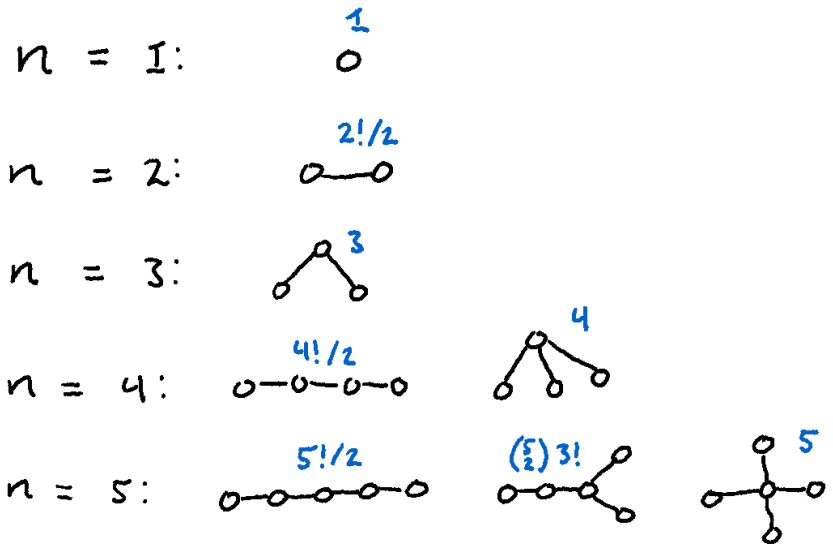
Vi har alltså lyckats räkna två specifika sorters träd. Kan vi räkna träd mer generellt?

Teorem 7. *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 7: 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.

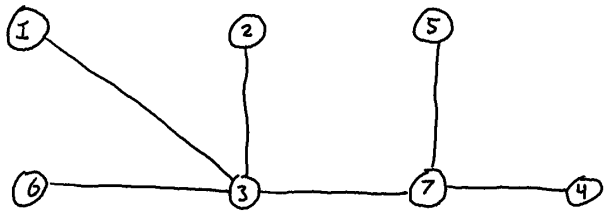
5. Vi får $\frac{5!}{2}$ av samma speglingssymmetri-skäl som innan, och för den tredje av våra grafer får vi 5 eftersom vi åter har en särskild nod och resten kan vi inte se skillnad på.

   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.[8] Vi letar upp det löv[9] som har lägst etikett, skriver dess etikett som första bokstav i vårt ord, och tar sedan bort noden ur trädet.



Figur 8: 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.

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

Det tar en stund att förstå vad den här algoritmen faktiskt gör[10], men efter en stunds kontemplation ser man att vad den gör är att

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

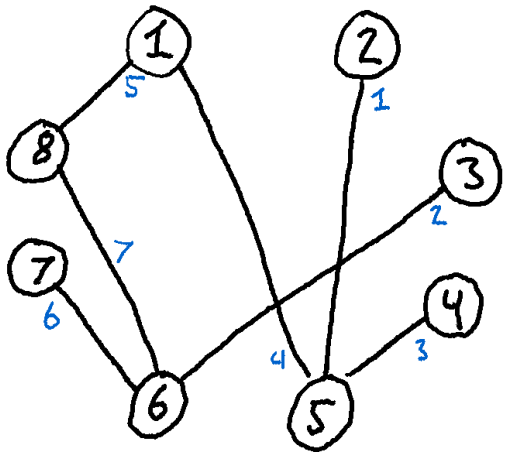
[9] Strikt sett har vi hittills bara definierat *löv* för rotade träd – och de träd vi studerar här har ju ingen rot-nod. Med "löv" menar vi här "nod med bara en granne".

[10] 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.

Algoritm 1: Konstruktion av träd från Prüferkoder

Låt G vara en graf med n noder, med etiketter $1, 2, \ldots, n$, men utan kanter
$L_1 \leftarrow (1, 2, \ldots, n)$
$A_1 \leftarrow (a_1, a_2, \ldots, a_{n-2})$
for $t = 1, 2, \ldots$ **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
    $L_{t+1} \leftarrow L_t \setminus \{l\}$
    $A_{t+1} \leftarrow (A_t(2), A_t(3), \ldots, A_t(|A_t|))$
  end if
end for
return G

den lägger till kanterna i precis den ordning som de försvann när vi skapade Prüferkoden.



Figur 9: 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. □

Ett alternativt bevis av Cayleys formel

Det bevis vi just gav av Cayleys formel är visserligen elegant, men det är inte det enda beviset av denna sats. Det finns åtskilliga andra bevis – inte bara med hjälp av bijektioner, utan också med rekursioner, och ett klassiskt bevis som använder determinanter och Kirchhoffs matris-träd-sats.

Beviset vi skall ge nu använder inga avancerade metoder alls, inte ens en bijektion, utan är bara ett helt vanligt "räkna på två sätt"-bevis. I boken jag tog det från [11] beskrivs detta som "det vackraste beviset av dem alla", men personligen föredrar jag nog Prüfers bevis, som inte ens var med på deras lista.

För att kunna ge detta bevis behöver vi definiera några till koncept.

Definition 8. En *riktad* graf är en graf där varje kant har en utpekad start- och slutnod. Vi tänker oss kanterna som pilar som pekar från start till slut.

Det finns ett uppenbart sätt att omvandla ett rotat träd till ett *riktat* träd[12] genom att ge varje kant den riktning som pekar bort från roten, och vice versa kan vi, för ett riktat träd med konsistenta riktningar på kanterna göra om det till ett rotat oriktat träd.

Definition 9. En graf H med noder V_H och kanter E_H är en *delgraf* till en graf $G = (V_G, E_G)$ om $V_H \subseteq V_G$ och

$$E_H \subseteq \{\{u, v\} \in E_G : u, v \in V_H\}.$$

För att skapa oss en delgraf till G tar vi alltså någon delmängd till dess noder, och tar sedan någon delmängd av kanterna mellan dessa noder.

Om graferna är riktade kräver vi att de skall ha samma riktning i G som i H. Om de är etiketterade kräver vi att etiketterna skall matcha i H och G.

Definition 10. En *skog* är en graf sådan att varje sammanhängande komponent är ett träd. Eller ekvivalent kan vi säga att det är en graf utan cykler.

En *rotad* skog är en skog med ett val av rot för varje träd i skogen.

Alternativt bevis av Cayleys sats. Låt $\mathcal{F}_{n,k}$ beteckna mängden av rotade skogar med n noder, som består av k träd. Alltså är $\mathcal{F}_{n,1}$ antalet rotade träd – om vi kan räkna dem är vi klara, eftersom vi vet att det finns precis n val av rot, så $\frac{|F_{n,1}|}{n}$ är talet vi söker.

Vi säger att en skog F' *ligger i* en annan skog F om F' är en delgraf till F när vi betraktar dem som riktade träd.[13]

Nyckelidén i vårt bevis är *klyvande följder* av rotade skogar. Vi säger att en följd $F_1, F_2, \ldots, F_k$ av rotade skogar på n noder är *klyvande* om varje F_i består av i stycken träd, och F_i innehåller F_{i+1} för varje i.

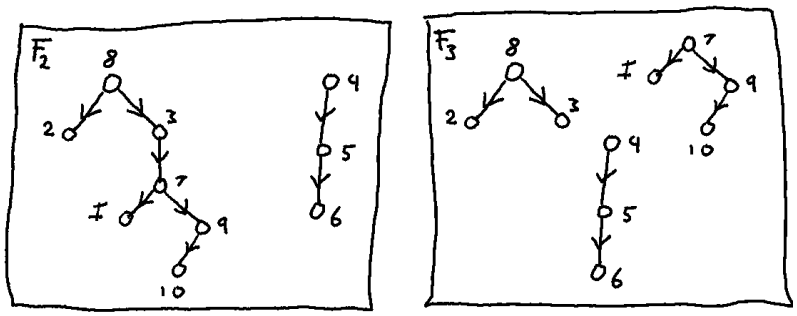
[11] Beviset är av Jim Pitman, men boken är *Proofs from the Book*. Den får detta namn från hur Erdős brukade påstå att Gud hade en bok som innehöll alla de vackraste bevisen. (Kanske en väldigt infantil version av de medeltida teologernas idéer om matematiken som Guds ordning i Hans skapelseverk?)

Hela idén om "boken" är egentligen lite märklig, med tanke på att Erdős var ateist – om än av typen som är arg på Gud för Hans påstådda ickeexistens.

Själva boken (den jag tog beviset ur, alltså) är, för övrigt, inte så särskilt bra, i min mening. Den gör bevisen mer komplicerade än de behöver vara, och har ett lite udda urval. Herren har definitivt en bättre bok, även om allvetande kanske får anses som fusk när man letar efter bevis.

[12] Alltså ett träd som har riktade kanter.

[13] Vi har ju inte definierat vad vi menar med att ett *rotat* träd skulle vara en delgraf till ett annat. För att vara tydliga betyder detta att vi *inte* kräver att roten till ett träd i F' också är rot till det större träd det ligger i i F.



Figur 10: Två rotade skogar F_2 och F_3, betraktade som riktade träd. Notera att F_3 ligger i F_2.

Vad som händer i den här definitionen är alltså att vi börjar med ett träd F_1, sedan klyver vi detta träd i två genom att ta bort en kant och får skogen F_2, och sedan klyver vi ett av träden i F_2 genom att ta bort ytterligare en kant och får skogen F_3 på tre träd, och så vidare.

Låt nu $F_k \in \mathcal{F}_{n,k}$ vara en fix skog, och beteckna

- antalet rotade *träd* som innehåller F_k med $N(F_k)$, och
- antalet klyvande följder som slutar med F_k med $N^*(F_k)$.

Notera nu att om vi tar $k = n$ finns det bara ett val för F_k, nämligen skogen av n stycken träd, där varje träd bara är en enda nod – och *varje* rotat träd på n noder innehåller denna skog. Alltså är $N(F_n)$ helt enkelt antalet rotade träd, vilket ju är vad vi vill räkna.

Så om vi kan hitta en formel för $N(F_k)$ är vi klara. Vi gör detta genom att räkna $N^*(F_k)$ på två olika sätt. Först går vi från F_k till F_1, och sedan i motsatt riktning.

Givet ett F_k, hur många val har vi av av F_{k-1}? Eftersom F_{k-1} ska ha ett träd färre, måste vi foga ihop två träd med en kant. Den här kanten kan utgå från vilken nod som helst, och skall träffa en av de $k-1$ rötterna till andra träd än det träd noden själv är i.[14] I Figur 10 kan vi till exempel få F_2 från F_3 genom att rita dit en kant från 3 till 7, som är roten för sitt träd.

Vi kan alltså välja F_{k-1} givet F_k på $n(k-1)$ sätt. Precis samma argument ger att vi kan välja F_{k-2} givet F_{k-1} på $n(k-2)$ sätt, och så vidare. Så totalt kan vi välja vår följd $F_k, F_{k-1}, \ldots, F_1$ på

$$N^*(F_k) = n^{k-1}(k-1)!$$

sätt.

Så till den andra riktningen – givet F_k finns det, per definition, precis $N(F_k)$ val av F_1, eftersom F_1 skall vara något rotat träd som innehåller F_k. Vi ser enkelt att det är precis $k-1$ kanter som ligger i F_1 men inte i F_k – när vi räknade i motsatta riktningen är det dessa vi lade till.

Tar vi bort en av dessa kanter klyver vi F_1 och får en skog F_2, tar vi bort en till klyver vi ytterligare ett träd och får F_3, och så vidare – och

[14] Den måste träffa en rot för att riktningen på kanterna skall förbli konsekvent – om den träffade en ickerot hade vi fått en nod som är slutnod för två kanter, och alltså har två föräldrar, vilket vi inte tillåter.

varje klyvande följd $F_1, F_2, \ldots, F_k$ med fixt F_1 och F_k kan fås på detta vis.

Uppenbarligen är antalet sätt att välja en ordning att ta bort de $k-1$ kanterna i precis $(k-1)!$, så vad vi har visat är att

$$N^*(F_k) = N(F_k)(k-1)!$$

vilket tillsammans med vårt andra sätt att räkna $N^*(F_k)$ ger oss att

$$N(F_k) = n^{k-1}.$$

Vi observerade innan att $N(F_n)$ är lika med det totala antalet rotade träd, och eftersom det finns n sätt att välja roten till ett träd är talet vi letar efter alltså

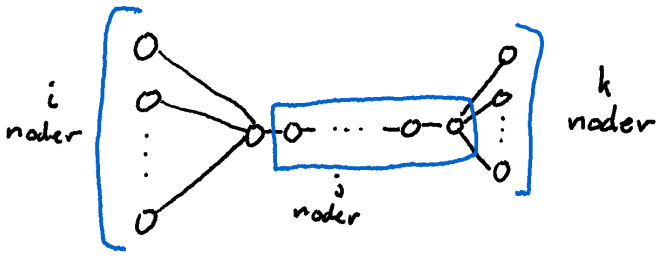
$$\frac{N(F_n)}{n} = \frac{n^{n-1}}{n} = n^{n-2}$$

såsom önskat. □

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 11: Skiss av ett oetiketterat träd.

Hur många olika sätt finns det att sätta etiketter på detta träd?[15] Ge en formel som gäller för alla $i, j, k = 0, 1, 2, \ldots$.[16]

Övning 3. Vi visade i Proposition 5 att antalet rotade ordnade binära oetiketterade träd på $n-1$ interna noder ges av Catalantalen, och i Proposition 6 att antalet rotade ordnade oetiketterade träd på $n+1$ noder ges av Catalantalen. Alltså måste det specifikt finnas lika många av dessa två olika sorters träd.

Finn en explicit bijektion mellan dessa mängder av träd.[17]

Övning 4. Rita det etiketterade trädet som har Prüferkod 1273262.

Övning 5. *En övning för dig som kan programmera.*[18] Implementera vår algoritm för att omvandla en Prüferkod till ett etiketterat träd i faktisk

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

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

[17] Om ni vill ha en ledtråd kan ni leta upp hur programspråket Lisp representerar träd.

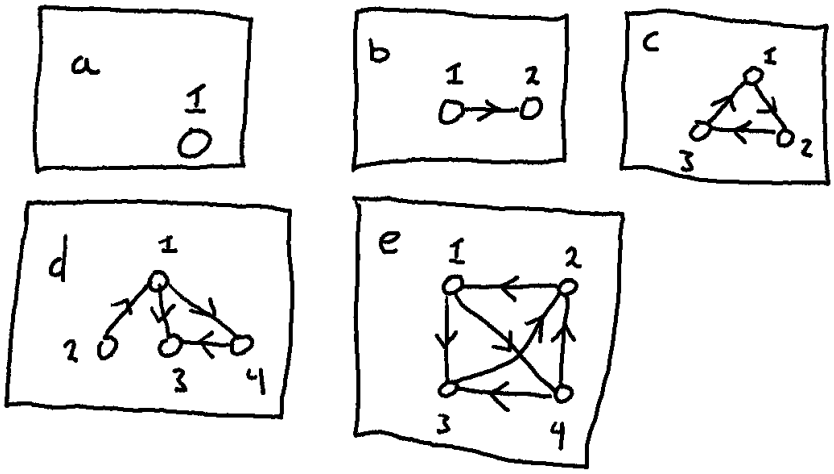
[18] För inlämningsuppgiften i kursen är denna uppgift frivillig om ni inte har någon i er grupp som kan programmering. Om ni genomför den, vänligen använd inte Matlab, eftersom jag inte kan köra sådan kod. Mathematica är okej.

kod. Koden skall ge ett lämpligt grafobjekt som output, och en bild av grafen.[19]

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

[19] 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.

Övning 6. Betrakta Figur 12, med fem olika riktade etiketterade grafer ritade. Vilka grafer är delgrafer till vilka?



Figur 12: Fem olika riktade etiketterade grafer, *a* till *e*.

Föreställ er nu att vi glömmer bort riktningen på alla kanterna, så att graferna blir *oriktade* etiketterade grafer. Hur förändras era svar?

Om vår glömska fortsätter, och vi glömmer också etiketterna, hur ändras era svar? I det här fallet kan vi till och med ha att en graf är en delgraf till en annan på flera olika sätt – till exempel finns det nu två olika delgrafer till b som bägge är a (ta en av bs två noder), och c har tre delgrafer som är b (ta två av noderna och kanten mellan dem). Ange för varje par av grafer hur många olika sätt den ena är en delgraf till den andra, som oetiketterade grafer.

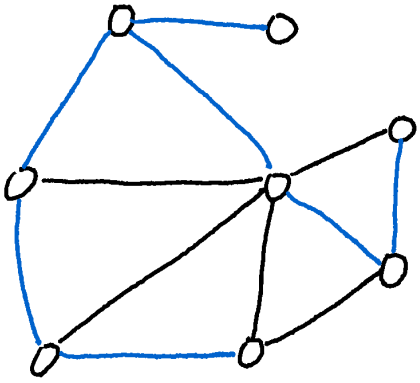
Övning 7. För varje n betecknar vi med K_n den *fullständiga grafen* på n noder, vilken är grafen med n noder med etiketter från 1 till n och varje möjlig kant närvarande. Så $K_n = \left([n], \binom{[n]}{2}\right)$, i formell notation.

För en graf G säger vi att ett träd H är ett *uppspännande träd* för G om

- H är en delgraf till G,
- $V_H = V_G$, så varje nod i G är med i H,
- och H är ett träd.

Hur många uppspännande träd finns det för K_n?[20]

[20] Ledtråd: Det här är en sak vi redan studerat, bara formulerad på ett annorlunda sätt. Fundera på vad det verkligen betyder att vara ett uppspännande träd för K_n.



Figur 13: En graf G med kanter i svart och blått. Tar vi enbart de blåa kanterna, och alla noder, får vi en delgraf H som är ett uppspännande träd för G.