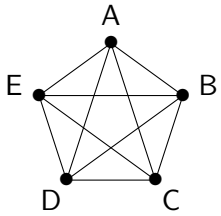
Grafteori

Marco Kuhlmann och Victor Lagerkvist

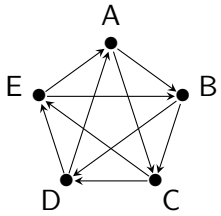
Grafteori är det område inom matematiken som undersöker egenskaper hos grafer. Inom grafteorin har begreppet *graf* en annan betydelse än ”graf till en funktion”.

5.01 En **graf** är en struktur av punkter som är sammanbundna med streck. Punkterna kallas **noder** och strecken kallas **bågar**. Grafen till höger har 5 noder och 10 bågar. Bågar som korsar varandra har ingen förbindelse med varandra. Noder kallas även för *hörn* och bågar kallas för *kanter*.



5.02 En vanlig formell definition av en graf är att betrakta såväl noder som bågar som *mängder*, och en graf kan då definieras som en mängd med noder, V, och en mängd med bågar, E, där $\{A, B\} \in E$ om det existerar en båge mellan noderna $A, B \in V$. Namnvalen kommer från engelska där noder vanligtvis *vertices* och bågar för *edges*. Exempelvis kan grafen i föregående exempel betraktas som en graf över nodmängden $\{A, B, C, D, E\}$ och med bågar $\{\{A, B\}, \{A, C\}, \{A, D\}, \{A, E\}, \{B, C\}, \{B, D\}, \{B, E\}, \{C, D\}, \{C, E\}, \{D, E\}\}$.

5.03 I en **riktad graf** har bågarna en riktning som indikeras med hjälp av pilar. En båge går alltså inte *mellan* två noder utan *från* en viss nod *till* en viss annan nod. I grafen till höger finns en båge från A till B men ingen båge från B till A; en sådan båge skulle kräva en pil åt andra hållet.



5.04 För att specificera att en graf över en nodmängd V är riktad måste bågmängden E definieras om. För alla A, B där A har en båge till B behöver vi en notation för att förtydliga detta samband. Notera att det i detta fall inte är lämpligt att betrakta en båge som en mängd $\{A, B\}$ eftersom vi i en mängd inte har någon inbördes ordning mellan elementen, och kan således inte utläsa om A har en båge till B, eller om B har en båge till A, utifrån $\{A, B\}$. Här är det i stället vanligt att använda den så kallade *tupelnotationen* och betrakta en båge mellan A och B som ett par (A, B), och mängden med bågar kan då definieras som en mängd med tupler över nodmängden V. Notera att en bågmängd nu kan betraktas som en binär *relation* mellan noder och kan således användas för att utläsa vilka noder som är relaterade till varandra.

5.05 Anta att du i ett datorprogram vill använda dig av en graf för att modellera någon form av data. Ett konkret exempel skulle kunna vara ett datorspel bestående av en karta som representerar spelvärden, som internt kan modelleras som en graf där viktiga punkter är noder, och som har en båge mellan sig om det är möjligt att röra sig mellan de två punkterna. Hur kan en sådan graf representeras som en datastruktur? Utgår vi ifrån den formella definitionen har vi redan en rimlig representation: har vi en datastruktur som representerar en mängd kan en graf kan enkelt representeras som två mängder, en mängd för noderna och en mängd för bågarna. En annan vanlig representation är att koda om grafen som en *matris*, en flerdimensionell array, där ett element (i, j) i matrisen är 0 om nod i inte har en båge till nod j, och 1 om det finns en båge mellan i och j. Denna representation kan tyckas krångligare än mängdrepresentationen men har vissa eleganta fördelar. Exempelvis kan man genom att multiplicera (med så kallad matrismultiplikation) matrisen med sig själv utläsa vilka noder i grafen som bildar en *triangel*: tre noder där varje inbördes par av noder har en båge mellan sig.

Grundläggande koncept

5.06 Antalet bågar som går in till en nod N, deg(N), kallas för nodens **grad** (ett annat begrepp är *valens*). Vad är gradantalet för noderna i grafen på den första sidan i manuskriptet?

För en så kallad *ögla* (en båge som börjar och slutar i samma nod) ska man räkna med bågens båda ändar.

5.07 Vi bevisar ett enkelt resultat kring grader:

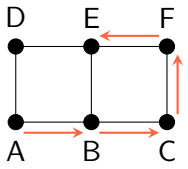
Lemma 5.1 Summan över alla gradtal i en graf är ett jämnt tal.

Bevis Vi använder en teknik som kallas för *double counting*. Den handlar om att räkna samma sak på olika sätt. Varje båge mellan två noder u och v bidrar med 2 till gradtalssumman, dels via u och dels via v. Mera formellt kan vi skriva följande:

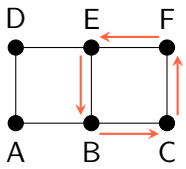
$$\sum_{\text{nod } u} \deg(u) \;=\; \sum_{\text{båge } e} 2$$

Den sista termen är lika med två gånger antalet bågar och därmed alltid ett jämnt tal.

5.08 En **väg** är en vandring längs bågarna i en graf som inte passerar någon båge mer än en gång. För att ange en väg kan man lista de noder som man besöker längs vägen. Ett exempel på en väg i grafen till höger är A–B–C–F–E.

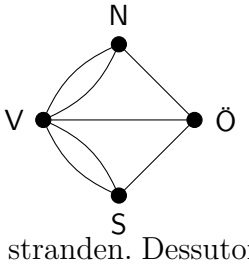


5.09 En **cykel** (ett annat begrepp är *krets*) är en sluten väg, dvs. en väg som leder tillbaka till startnoden (och passerar minst en båge). Ett exempel på en cykel i grafen till höger är är B–C–F–E–B.

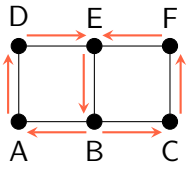


Eulergrafer

5.10 **Königsbergs sju broar.** Staden Königsberg (nuvarande Kaliningrad) låg vid bägge sidor av floden Pregel samt på två öar mitt i floden: en mindre västlig ö, som var stadens centrum, och en större östlig ö. Från den västliga ön gick två broar till den norra stranden och två broar till den södra stranden. Från den östliga ön gick en bro till den norra stranden och en bro till den södra stranden. Dessutom fanns en bro mellan öarna. Stadens invånare försökte på sina söndagspromenader att gå genom staden på ett sådant sätt att de passerade varje bro exakt en gång, men ingen hade någonsin lyckats med detta. Frågan var alltså: ”Finns en promenadväg som passerar varje bro exakt en gång?” Den schweiziske matematikern Leonhard Euler (1707–1783) visade år 1735 att svaret på denna fråga är ”nej”. Hans arbete brukar anses som ett av de första arbetena inom grafteori.



5.11 En **Eulerväg** är en väg som passerar varje båge i grafen exakt en gång. Ett exempel på en Eulerväg i grafen till höger är B–C–F–E–B–A–D–E. En sluten Eulerväg kallas för **Eulercykel**. En **Eulergraf** är en graf som innehåller en Eulercykel.



5.12 Vi kommer nu att följa Eulers resonemang.

Lemma 5.2 En graf med en Eulerväg innehåller högst två noder med udda gradtal.

Bevis Antag att grafen innehåller en Eulerväg P och låt v vara en godtycklig nod i grafen. Vi betraktar två fall:

1. Nod v ligger inte på P. Då kan v inte ingå i någon båge, eftersom P passerar alla bågar. Detta innebär att v har grad noll.

2. Nod v ligger på P men är inte den första eller sista noden på denna väg. Då gäller att man rör sig mot v lika många gånger som man rör sig bort från v när man följer P. Detta innebär att graden för v är ett jämnt tal.

I båda dessa fall är alltså graden för v ett jämnt tal. Det enda fallet som vi inte undersökt är om v är startnod eller slutnod i P. Med andra ord kan endast startnoden och slutnoden i P och därmed högst två noder ha udda gradtal.

5.13 Utifrån lemma 5.2 kan vi dra slutsatsen att det inte finns någon promenadväg genom Königsberg som passerar varje bro exakt en gång – i Königsberggrafen har nämligen alla fyra noder udda gradtal.

5.14 När grafen har en Eulercykel gäller en starkare version av lemma 5.2:

Lemma 5.3 En graf med en Eulercykel innehåller inga noder med udda gradtal.

Bevis I en cykel är startnoden och slutnoden identiska. Då gäller samma argument som vi använde i fall 2 i beviset av lemma 5.2 även för denna nod.

5.15 Lemma 5.3 ger ett *nödvändigt* villkor för att en graf ska kunna vara en Eulergraf. Euler påstod, men bevisade inte, att denna egenskap även utgör ett *tillräckligt* villkor för att en graf ska kunna vara en Eulergraf:

Lemma 5.4 En sammanhängande graf är en Eulergraf om och endast om den inte innehåller noder med udda gradtal.

Bevis Den ena delen av detta lemmat bevisade vi i beviset av lemma 5.3. Den andra delen har ett elegant men mera omfattande bevis som vi inte går igenom på denna kurs. Det publicerades först (posthumt) år 1873 av den tyske matematikern Carl Hierholzer (1840–1871).

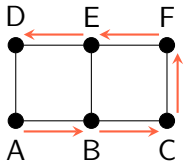
5.16 En graf kallas **sammanhängande** om det finns en väg mellan varje par av noder. Notera att noderna inte behöver vara olika; från en nod till sig själv finns det ju alltid den ”triviala” vägen som inte innehåller några bågar alls.

Hamiltongrafer

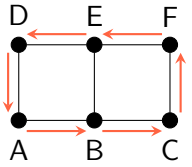
Utifrån lemma 5.4 är det väldigt lätt att testa om en graf är en Eulergraf: Vi går igenom alla noder i grafen. Om det finns en nod med udda gradtal så är svaret ”nej”, annars är svaret ”ja”. I nästa avsnitt kommer vi att studera en egenskap hos grafer där det är väldigt svårt att avgöra om en given graf har denna egenskap.

Vi börjar med en definition:

5.17 En **Hamiltonväg** är en väg som besöker varje nod i grafen exakt en gång. Ett exempel på en Hamiltonväg i grafen till höger är A–B–C–F–E–D. Notera att denna väg är inte en Eulerväg, eftersom den inte besöker bågarna B–E och A–D.



5.18 En sluten Hamiltonväg kallas för **Hamiltoncykel**. Ett exempel på en Hamiltoncykel i grafen till höger är A–B–C–F–E–D–A. En **Hamiltongraf** är en graf som innehåller en **Hamiltoncykel**.

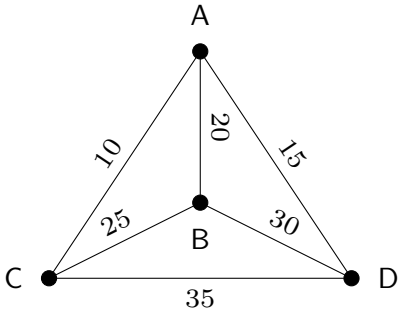


5.19 Ett klassiskt grafteoretiskt problem som har med Hamiltongrafer att göra är **handelsresandeproblemet** (eng. *travelling salesman problem*). En handelsresande ska besöka ett antal städer och sedan återvända till sitt hem. Om man modellerar städerna som en fullständig viktad graf (en graf som innehåller bågar mellan alla noder och där man för varje båge angett ett numeriskt värde, i det här fallet avståndet mellan två städer) motsvarar handelsresandeproblemet uppgiften att hitta den Hamiltoncykel som ger kortast reslängd.

5.20 Handelsresandeproblemet är ett så kallat NP-hårt problem och därmed ett av de beräkningsmässigt svåraste problemen inom datalogin. Man misstänker att det inte finns någon effektiv lösning för problemet.

5.21 En metod som man kan använda för att approximera en lösning för handelsresandeproblemet är **närmaste granne-metoden**. Enligt denna metod väljer man i varje steg den båge som har minst vikt (den stad som ligger närmast).

Antag att vi vill tillämpa närmaste granne-metoden på grafen som visas till höger. Vi börjar i stad A och vill besöka alla städer innan vi återvänder till A. Närmaste granne-metoden ger oss följande resväg: A–C–B–D–A. Det totala avståndet summan av avstånden för de bågar som vi passerade: $10 + 25 + 30 + 15 = 80$.



Närmaste granne-metoden hittar inte nödvändigtvis en optimal väg, och i vissa fall hittar den inte någon väg alls.

Träd

5.22 En (oriktad) graf som är sammanhängande och inte innehåller några cykler kallas för ett **träd**. Dessa strukturer spelar en stor roll inom datalogin.

5.23 De två karakteristiska egenskaper hos ett träd – att de är sammanhängande och inte innehåller några cykler – hänger ihop på ett särskilt sätt. Detta kan man se när man läser bevisen på de följande två påståendena:

Lemma 5.5 Om man lägger till en båge till ett träd får man en cyklisk graf.

Bevis Låt T vara ett träd. Vad händer om vi lägger till en båge till T? Låt oss skriva bågen som v–w. Eftersom ett träd är sammanhängande vet vi att det i T finns en väg som börjar i v och slutar i w redan innan vi lägger till den nya bågen. Om vi vandrar längs denna väg och som ett sista steg följer den nya bågen kommer vi tillbaka till v. Alltså har vi hittat en cykel.

Lemma 5.6 Om man tar bort en båge från ett träd blir grafen osammanhängande.

Bevis Låt T vara ett träd. Vad händer om vi tar bort en båge v–w från T? Eftersom ett träd är acykliskt vet vi att bågen v–w är den enda vägen mellan v och w. (Om det nämligen fanns en annan väg skulle vi tillsammans med bågen v–w ha en cykel.) Om vi nu tar bort bågen v–w finns det alltså *ingen* väg mellan v och w längre; då blir alltså T osammanhängande.

5.24 Träd är intressanta inte bara i sig, utan även för att analysera andra typer av grafer. Ett **uppspännande träd** för en graf G innehåller alltså samma noder som G och en delmängd av dess bågar.

5.25 Det finns två enkla algoritmer (procedurer) för att hitta ett uppspännande träd för en given graf G:

1. Låt H vara en kopia på G. Testa om H är ett träd. Om den inte är det, ta bort en godtycklig båge från H, men se till så att inte H blir osammanhängande.

2. Låt H vara grafen som har samma noder som G, men inga bågar. Testa om H är ett träd. Om den inte är det, lägg till en godtycklig båge från E till H, men se till att inte skapa cykler. Denna algoritm kallas för **Kruskals algoritm**.

I bägge fall så kommer H så småningom bli ett spännande träd för G.

5.26 Ett **minimalt uppspännande träd** för en viktad graf G är ett uppspännande träd för G som har minimal viktsumma bland alla uppspännande träd för G. För att hitta ett minimalt uppspännande träd kan man använda Kruskals algoritm där man i varje steg lägger till den billigaste bågen (utan att skapa cykler).