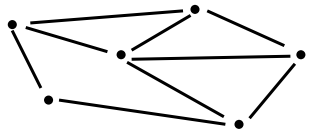
KAPITEL 7 - GRAFTEORI

1. Grafer: strukturer på mängder

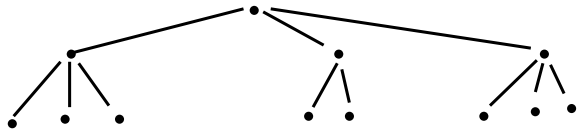
Ingenjörskonsten inom datalogin kan fundamentalt betraktas som tillämpad diskret matematik. En av de bästa illustrationerna av det är vi när studerar situationer som involverar flera entiteter och vi vill arbeta med att dessa entiteter hänger ihop på olika sätt. Ordet "entitet" är bara en allmän teknisk term som används för att referera till vad vi arbetar med, det kan betyda vad som helst, på samma sätt som ordet "element", så grafer hjälper oss att studera strukturer som uppstår på mängder av vissa speciella element.

Entiteterna/elementen kan vara till exempel vara kunder i en databas och produkter som de vill köpa. Relationerna kan då vara att en viss kund vill köpa en viss produkt. Och från kapitlet om relationer vet vi att en relation modelleras och defineras med mängder. Vi kan hitta på andra exempel: entiteterna kan vara processer i ett operativsystem och relationen kan vara att en process har skapat en annan process. En annan liknande situation kan vara att vi vill skriva ett program som hanterar problemställningar kring om olika orter som är förbundna med varandra via vägar. Entiteterna här är då orterna och en relation mellan två entiteter är att det finns en väg mellan orterna. Vi kan illustrera det genom att ge en bild där orterna symboliseras av punkter och två punkter har ett streck mellan sig innebär precis att det finns en väg mellan dessa två orter.



Vi observerar att ovanstående bild skulle kunna representera något helt annat: till exempel skulle varje punkt kunna vara en dator i ett nätverk och strecken symboliserar då att två datorer i nätverket har direkt kommunikation med varandra. En *graf* är då en abstrakt modell (som vi snart ska precisera) som kan användas för att modellera en situationer som beskrivits ovan. För att bygga upp en bra terminologi kring det här ska vi kalla punkterna för *hörn* och strecken mellan punkterna/hörnen för *kanter*.

Vi studerar en annan bildmässig representation av en graf:



Det här är en graf med 12 hörn med diverse kanter utritade. Grafen har en speciell egenskap: mellan två olika hörn finns högst en kant. Den här grafen är dessutom av en typ som vi senare kommer att kalla *träd* och sådana grafer kan ritas på ett hierarkiskt sätt: ett visst hörn här i grafen ovan är utvalt och vi ritat det högst upp och i sådana här skisser kallar vi det utvalda hörnet för trädets *rot*. Vi har också en underförstådd riktning i grafen, och hörn som ligger under andra hörn kallas då dessa hörns *barn*. Den här typen av träd skulle då kunna användas för att modellera situationen med processer som skapar andra processer, (sådana processer brukar då kallas föräldra- respektive barnprocessser) men vi ser också att samma typ av träd skulle kunna användas för att modellera kataloger i en filstruktur. Eller så kan vi använda träd för att beskriva hur komponenter i ett mjukvarusystem hänger samman. Som vi ser är möjligheterna väldigt vittgående och vi ska nu införa den formella definitionen av grafbegreppet.

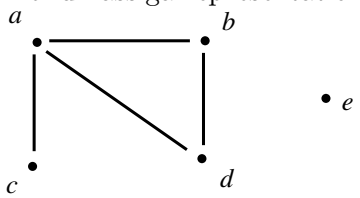
Definition: En *graf* är ett par av två mängder (V, E), där $V \neq \emptyset$ och E är en mängd av mängder av hörn med två olika element valda ur V. Elementen i V kallas grafens *hörn* och elementen i E kallas grafens *kanter*. Om $e = \{v, w\}$ är en kant (där $v, w \in E$) så sägs e *förena* hörnen v och w och dessa hörn sägs då vara *grannar*. Hörnen v och w kallas också *es ändpunkter*.

Antalet kanter som utgår från ett visst hörn kallas det hörnets *gradtal* eller *grad* och det betecknas $deg(v)$, där v är hörnet i fråga. Ett hörn kallas *jämnt* (respektive *udda*) omm den har ett jämnt (respektive udda) gradtal. Ett hörn med graden 0 sägs vara *isolerat*.

En graf (V, E) sägs vara *ändlig* om V är en ändlig mängd. (Alla grafer kommer hädanefter att vara ändliga i den här framställningen.) När vi säger "Låt $G(V, E)$ vara en graf" så inför vi en graf som vi betecknar med G som har hörnmängd V och kantmängd E.

Skrivsätt: Ibland förkortar vi notationen och skriver vw istället för $\{v, w\}$ när vi vill beteckna ett hörn i en graf och inte vill skriva det längre uttrycket $e = \{v, w\}$.

Vi använder alltså mängdbegrepp för att formalisera precis vad en graf är och vi ska nu se hur det hänger ihop med bildrepresentationen av en graf. Studera grafen $G(V, E)$, där $V = \{a, b, c, d, e\}$ och $E = \{ab, ac, ad, bd\}$. Den bildmässiga representationen anges nedan.



Vi ser att hörnet e är ett isolerat hörn, att a är granne till b, c, d men att b inte är granne till c. Gradtalen i grafen ges av $deg(a) = 3$, $deg(b) = 2$, $deg(c) = 1$, $deg(d) = 2$ och $deg(e) = 0$. Det betyder att mängden av jämna hörn i grafen är $\{b, d, e\}$ och mängden av udda hörn är $\{a, c\}$.

En graf är alltså ett par av två mängder (V och E). Det betyder att grafbegreppet är mycket generellt och där kan grafteorin ge oss resultat som kan tillämpas i många olika situationer. Som vi så ovan kunde grafer användas för att representera vitt skilda problemställningar. Rubriken för det här avsnittet motiveras alltså genom att grafer blir ett sätt att införa strukturer på mängder: vi har en mängd av entiteter som vi vill studera. Dessa entiteter kallas då *hörn* och strukturen beskrivs genom att vi anger mängden av kanter.

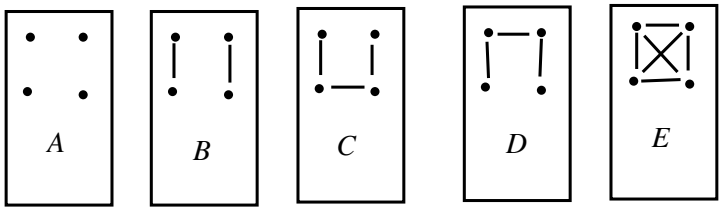
Eftersom grafer då kan anses som mängder med strukturer så kan de tidigare införda mängdbegrepp också användas på grafer. När vi har en graf G så kan vi alltså tala om att lägga till eller ta bort hörn eller kanter för att skapa större eller mindre grafer. Vi talar dock inte om "den tomma grafen", vi har i definitionen krävt att vi åtminstone ska ha ett hörn. Men grafer har ändå mycket likheter med mängder, vi har där begreppet *delgraf* som vi nu ska införa.

Definition: Låt $G(V, E)$ vara en graf. En graf $G_1(V_1, E_1)$ kallas då en *delgraf* av G om och endast om $V_1 \subseteq V$ och $E_1 \subseteq E$. Vi skriver detta $G_1 \subseteq G$. (Vilket då innebär att vi med den här definitionen inför en ny betydelse hos tecknet $\subseteq$.)

I samband med den här definitionen är det naturligt att införa "den fullständiga grafen":

Definition: Låt V vara en mängd med n element. Den *fullständiga grafen på n hörn* betecknas med K_n och definieras då som grafen (V, E) där E är mängden av *alla* kanter som går att bilda mellan hörn i V.

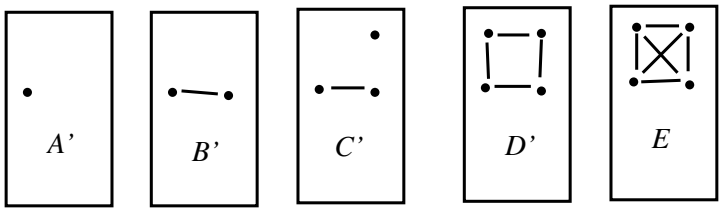
Så om vi har en ängd V av n element så kan vi bilda många olika grafer genom att associera V med en mängd av kanter. Vi skulle kunna formera $(V, \emptyset)$, vilket förstås skulle vara en väldigt tråkig graf eftersom det inte skulle finnas några kanter, men till den här grafen skulle vi gradvis kunna lägga till kanter och få större och sörre grafer. Till slut har vi lagt till alla mjliga kanter och då har vi nått K_n. Vi illustrerar detta genom att studera en tänkt mängd med fyra element. Om vi lägger till kanter och skapar olika grafer G, har för alla dessa kanter delgrafsrelationen $(V, \emptyset) \subseteq G \subseteq K_4$. I figuren nedan ger vi fem sådana grafer, den längst till vänster är $(V, \emptyset)$, det vill säga den har inga kanter (vi betecknar också den med A), medan den längst till höger är K_4, det vill säga den har alla möjliga kanter som den kan ha (vi betecknar också den med E). Mellan dessa ytterligheter har vi de tre mellanliggande graferna B, C, D:



Om vi betraktar dessa grafer så ser vi att följande delgrafsrelationer är uppfyllda: $A \subseteq B \subseteq C \subseteq E = K_4$ och $A \subseteq B \subseteq D \subseteq E = K_4$. Observera dock att vi inte har $C \subseteq D$ eller $D \subseteq C$ eftersom kantmängderna hörande till C och D inte uppfyller någon delmängdsrelation.

Definitionen av delgraf säger inte att antalet hörn måste överenstämma, men ett självklart krav är att en delgraf måste ha färre antal hörn än den graf som den är delgraf av. Vi kan illustrera det genom att modifiera

det föregående exemplet och studera andra delgrafer av K_4, och här ersätter vi graferna A, B, C, D med andra liknande grafer men med avtagande antal hörn. Vi kallar de nya graferna A', B', C', D':

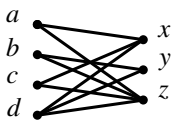


I den här nya situationen är varje graph till vänster delgraf av den till höger så att vi kan skriva $A' \subseteq B' \subseteq C' \subseteq D' \subseteq E = K_4$.

I det här läget kanske läseren ser väldigt stora likheter med delmängdsrelationen. Och faktiskt kan vi visa att mängden av alla grafer utgör en partiellt ordnad mängd under delgrafsrelationen. Läsaren uppmanas att genomföra det beviset.

1.1. **Bipartita grafer.** En vanlig situation som uppkommer i databastillämpningar är att vi vill studera relationer mellan entiteter ur en viss kategori (kanske kunder) och hur dessa entiteter relaterar till entiteter i en annan kategori (kanske produkter som kunderna vill köpa, hyra eller låna). Med dessa få ord har vi genast beskrivit stora mängder av vårt samhälles administrationsbehov och vi kommer i detta avsnitt studera en viss typ av grafer som kan modellera dessa tillämpningar: de bipartita graferna.

För att beskriva vad en bipartit graf är ska vi beskriva ett exempel. Vid ett universitet ges många kurser och många studenter går många kurser. Vi säger att studenter och kurser har ett så kallat "många-till-många"-förhållande, det anser vi som det mest komplicerade förhållandet som kan råda mellan entiteter av två olika slag (här studenter och kurser). Om vi antar att det finns fyra studenter, a, b, c, d och tre kurser, x, y, z, så skulle en sådan här situation kunna beskrivas av följande graf.



Grafen uttrycker alltså att a går kurserna x och z, b går y och z, c går x och z och d går alla kurser, x, y och z.

Det här är en så kallad *bipartit graf* som alltså beskriver relationer mellan två kategorier av element. Vi tar en formell definition på vad en bipartit graf är.

Definition: En *bipartit graf* är en graf vars hörnmängd kan partitioneras i två klasser, V_1 och V_2 på ett sådant sätt att om vw är en godtycklig kant så gäller att $v \in V_1$ och $w \in V_2$ eller det omvända $w \in V_1$ och $v \in V_2$. En *fullständig bipartit graf*, G, är en bipartit graf som har alla möjliga kanter, det vill säga (med notationen V_1 och V_2), för alla möjliga par av hörn $v \in V_1$ och $w \in V_2$ så är vw en kant i G. Den *fullständiga bipartita grafen på m respektive n hörn* defineras som den bipartita graf med hörnklasser V_1 respektive V_2, där $|V_1| = m$ och $|V_2| = n$. Den betecknas med $K_{m,n}$.

(I exemplet ovan hade vi $V_1 = \{a, b, c, d\}$ och $V_2 = \{x, y, z\}$.)

Övningar

7.1.1 Betrakta nedanstående konstruktioner. Vilka av dem är grafer och vilka är inte grafer?

(a) $(\{a, b, c, d\}, \{ad, bc, bd, dc\})$ *(b)* $(\{a, b, c, d, e\}, \{\{a\}, \{a, d\}, \{b, c\}\})$ *(c)* $(\{a, b, c, d\}, \{\{a, e\}, \{a, d\}, \{c, d\}\})$

Rita figurer för allihop som illustrerar de grafer som ges men rita också figurer även om det som ges inte är grafer och illustrera själva problemet: varför är det som ges inte en graf? (För de konstruktioner som inte är grafer är det förstås kanske inte helt klart hur de ska ritas, men rita någonting så resonerar vi kring det tillsammans.)

I Finans och Bogart-Drysdale-Steins böcker görs en lite annorlunda definition av grafbegreppet – de introducerar begreppet "simple graph", alltså "enkel graf" för det som vi kallar graf. Deras grafbegrepp är alltså lite

bredare än vårat och de tillåter kanter som går från ett hörn tillbaka till samma hörn – det kallas då en ***loop*** *– och vidare tillåter de också att två hörn kan ha flera olika kanter som förbinder dessa hörn. Vi kommer att kalla sådana konstruktioner för* ***pseudografer*** *och vi kommer att introducera detta begrepp i nästa avsnitt, men vi gör dessa kommentarer här för att ni lättare ska kunna arbeta med övningarna i Finan och Bogart-Drysdale-Steins böcker. De använder en* ***datalogisk*** *terminologi, vi använder en* ***matematisk*** *terminologi.*

Finan: Exempel 34.3, Exempel 34.4, Exempel 34.5, Exempel 34.6, Problem 34.1.

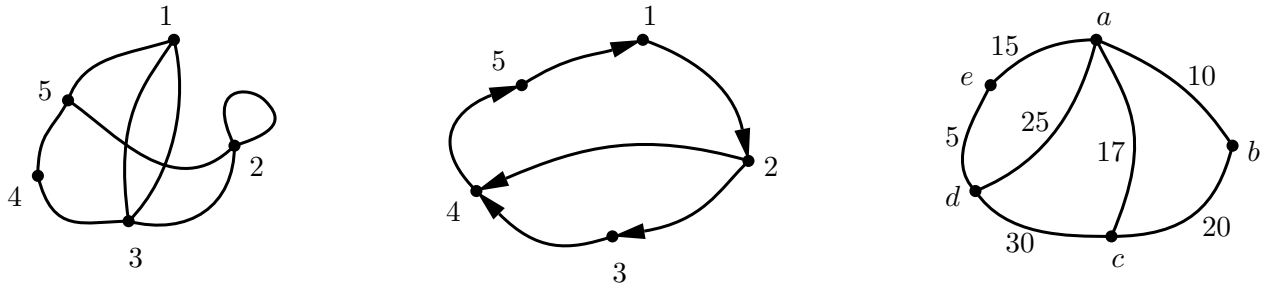
Bogart-Drysdale-Stein: Exercise 6.1-1, Exercise 6.1-2, Exercise 6.1-3, Problem 3, Problem 10, Problem 11.

2. Pseudografer, riktade grafer, viktade grafer och grannmatriser

I definitionen av vad en graf är så krävs det att en kant ska vara en mängd av två olika hörn. Det här kravet gör definitionen av grafer ganska inskränkt och vi kan observera flera brister i definitionen:

(1) Vi kan inte ha en kant från ett hörn tillbaka till samma hörn.
(2) Två olika hörn kan inte vara förbundna med flera kanter, högst en kant mellan varje par av hörn.
(3) En kant kan inte modellera en *riktning*, det vill säga ange att ett av hörnen i en kant skulle bestämmas som starthörn och det andra hörnet som sluthörn.
(4) En kan kan inte förses med en kostnad eller vikt vilket är mycket användbart inom tillämpningar där kanterna representerar att två entiteter är förbunda över någon form av avstånd.

Följande konstruktioner kan alltså inte modelleras av vår nuvarande definition av grafbegreppet så som det ser ut just nu:



De här konstruktionerna kan inte modelleras av de nuvarande grafbegreppet men vi kan ändå tänka oss att de här tre skisserna kan vara modeller av viktiga ingenjörsmässiga tillämpningar. Skissen i mitten liknar till exempel ett tillståndsdiagram som kan beskriva funktionen hos ett så kallat sekvensnät inom digitaltekniken. En övergång från ett tillstånd till ett annat är en *riktad* process, vi har ett starttillstånd och ett sluttillstånd, men det finns ingenting i vår nuvarande definition av grafer som möjliggör en representation av detta. Skissen till höger kan sägs beskriva ett vägnät mellan fem orter betecknade med a, b, c, d, e och att två orter har en väg som förbinder dem illustreras av att det finns en kant mellan de hörn som representerar orterna. Och den mellanliggande kanten har då också ett associerat tal, kallat *vikt* eller kostnad som anger hur mycket det kostar att använda vägen. Kostnaden skulle helt enkelt kunna representera hur lång vägen är. (I skissen till höger har vi noterat hörnen med bokstäverna a, b, c, d, e för att inte blanda samman dem med kostnaderna som är associerade med kanterna.) Slutligen har vi skissen till vänster som inte kan representeras av en graf av två anledningar. Den första är att hörn nummer 2 har en kant som går tillbaka till hörn 2. Detta kallas en *loop* och gå alltså inte att representera med vår nuvarande definition av grafer eftersom en kant måste anges av två olika hörn. Den andra anledningen till att den vänstra skissen inte kan representeras är att mellan hörnen 1 och 3 förekommer två kanter.

Observera att i ovanstående stycke har vi använt orden ”hörn” och ”kant” även fast hela stycket poängterar att det vi beskriver tyvärr inte kan anses vara grafer. Vi har valt dessa ord för att skapa en tydlighet om vad vi skulle önska: vi *vill* kunna representera dessa ovanstående konstruktioner som grafer på ett noggrannt sätt. I det här avsnittet ska vi undersöka en del möjligheter till det och vi ska basera vår diskussion på figuren ovan med de tre typerna av skisser som vi skulle vilja formalisera som grafer.

2.1. Riktade och viktade grafer – grannmatrisen.

Vi börjar med att studera den mittersta skissen och den till höger. För att kunna representera situationen ska inför någonting som kallas för en *riktad* graf. Vi skulle kunna definiera en riktad graf genom att i den ursprungliga definitionen för graf helt enkelt bara byta ut definitionen av en kant så att en kant inte ges av en mängd av två hörn utan istället ges av ett ordnat par, (v, w), där v är det första hörnet, som vi skulle kunna kalla ”starthörn” och w är det andra hörnet som vi skulle

kunna kalla ”sluthörn”. Vi ska dock inte göra så här av en anledning som kommer att klarna så småningom. Vi ska gå en annan väg och införa ett helt annat sätt att representera grafer: via matriser. Vi inför detta genom att studera ett exempel.

Exempel: Bilda en matris baserad på den mittersta skissen på följande sätt: matrisen ska 5 rader och 5 kolonner. På rad i, i kolonn j ska det stå talet 1 om och endast om det finns en kant med starthörn i och sluthörn j. Om det inte finns någon kant mellan hörn i och j ska det stå en 0:a där. Nu har vi återigen använt orden ”hörn” och ”kant” trots att den skiss vi talar om ovan är till för att illustrera att det här skisser inte är grafer, men vi ser inte det som ett problem här. Den matris som uppkommer genom den här processen ser ut så här:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Den översta raden är rad 1 och beskriver alltså alla kanter som utgår från hörn 1. Det finns bara en kant och den går till hörn 2 alltså skrevs en 1:a in på rad 1, kolonn 2. På samma sätt utgör rad två motsvarande beskrivning av alla kanter som utgår från hörn 2. Det finns två kanter, en till hörn 3 och en till hörn 4 varför vi har 1:or i kolonn 3 respektive 4 på rad 2. De följande raderna skapas förstås på samma sätt.

Vi har nu skapat ett alternativt representationssätt för grafer. En matris som skapas på det sätt som illustreras i exemplet kallas för *grannmatrisen* och en grannmatris kan alltså specificera uppbyggnaden av en ändlig så kallat *riktad* graf där varje kant har ett starthörn och ett sluthörn. Det förutsätts givetvis att vi associerar de ingående hörnen med ett tal och antal rader i grannmatrisen blir då lika med antal hörn i den graf vi vill representera. Grannmatrisen är förstå även kvadratisk så antalet kolonner blir också lika med antalen hörn i grafen som vi vill representera.

Det fina är nu att vi kan inse att vi kan representera grafer enligt den ursprungliga definitionen också med grannmatriser. Dock behöver vi först bestämma oss för om vi vill anse att kanterna i en vanlig graf också ska vara riktade och om vi har vanlig graf där hörnen v och w är förbundna med en kant, vill vi då anse att detta innebär att denna kant ska vara likvärdig med att det finns två riktade kanter, $v \to w$ respektive $w \to v$, och att båda tillsammans utgör kanten $\{v, w\}$? Om vi går med på det så kan vanliga grafer representeras med grannmatriser och de blir då symmetriska matriser med 0:or i diagonalen och 1:or på de ställen som specificerar grafens kanter. Grannmatrisen blir då här symmetrisk eftersom vi inte har någon riktning hos kanterna.

I och med att vi infört grannmatrisen på det här sättet så har vi gjort en speciell manöver som ofta sker i matematiken: vi har utvidgat representationen av ett begrepp (här grafer) och skapat en mer generellt begrepp (här riktad graf) och på detta sätt har vi ökat möjligheterna att skapa tillämpningar av teorin. Det kommer att visa sig att grannmatrisen är användbar för att kunna avgöra om vi kan hitta en väg mellan två noder i en graf. (En väg är en följd av kanter som förbinder två givna noder.)

Vi går nu till den högra skissen. I diskussionen ovan infördes det som kallades för en *riktad* graf och riktningen hos alla kanter bestämdes av hur 1:orna som specificerade kanterna placerades i grannmatrisen. Med en stunds eftertanken inser vi dock att vi faktiskt skulle kunna skriva andra tal än 1:or i grannmatrisen och vi har en annan typ av utvidgning som är möjlig: på rad i och i kolonn j noteras *vikten* som är associerad med kanten som går från hörn i till hörn j. Noga taget ger detta upphov till en *riktad och viktad* graf men vi sk inte modellera så noggrant utan vi nöjer oss med att modellera en *viktad* graf där alltså kanterna inte har en riktning utan bara en vikt. Grannmatrisen för grafen i skissen till höger ovan får då följande utseende:

$$\begin{bmatrix} 0 & 10 & 17 & 25 & 15 \\ 10 & 0 & 20 & 1 & 0 \\ 17 & 20 & 0 & 30 & 0 \\ 25 & 0 & 30 & 0 & 5 \\ 15 & 0 & 0 & 5 & 0 \end{bmatrix}.$$

Vi skulle också kunna modellera viktade och riktade grafer genom att inte arbeta med symmetriska matriser men vi avstår från det i den här framställningen.

2.2. **Pseudografer.** Till sist uppstår till sist frågan om hur vi modellerar situationen där vi har flera kanter mellan två hörn. Vi skulle teoretiskt sett då kunna definiera en graf som en mängd hörn och ange kanterna

som en *följd* av tripplar (v, w, k), där v och w är hörn och k är kantens kostnad. En trippel är inte en mängd så vi med denna konstruktion ange alla typer av grafer som vi kan komma på. Men vi gör inte detta. Vi slappnar av definitionskravet en aning och konstaterar att det är möjligt att göra dessa och vi kommer att ge en framställning av den följande teorin som om dessa noggranna definitioner gjorts, men på grund av att notationen i bevisen kan bli övermäktig avstår vi från det. Vi litar på att det kommer att gå bra.

För ändå vara någotsånär medvetna om vad vi gör så inför vi ytterligare ett begrepp (som vi faktiskt inte preciserar, men som skulle kunna preciseras med den här konstruktionen med kanter som tripplar) för att täcka in situationen då vi har grafer som inte kan specificeras enligt den ursprungliga mängddefinitionen.

Informell definition: En graf som inte riktigt är en graf eftersom den har loopar eller flera kanter mellan två av dess hörn kallas en *pseudograf.*

Pseudografer med loopar *kan* representeras av en grannmatris som då får 1:or i diagonalen, men vi kan inte använda grannmatriser för att representera pseudografer med flera kanter mellan två av sina hörn.

2.3. **Sammanfattning.** Detta avsnitt har alltså resulterat i tre nya begrepp: pseudografer, riktade grafer och viktade grafer och ett nytt instrument: grannmatrisen. Tonen i avsnittet har varit informell på flera sätt, dels har vi infört grannmatrisen via exempel och inte teckat definitionen formellt sett, men den formella defintionen är inte svår att skapa. Vi har också sett mer informalitet i den informella definitionen av pseudograf.

Det är faktiskt så att vi till och med kan anse införandet av grannmatris som inte bara en alternativ *representation* av viktade och/eller riktade grafer, vi skulle till och med kunna anse det som en alternativ *definition* av hela grafbegreppet. Det är ytterligare en anledning till att vi inför den via exempel och inte på ett formellt sätt. Om vi skulle gjort formella definitioner baserade på grannmatrisen så skulle vi behövt formulera bevis för att säkerställa att den nya definitionen (med grannmatris) är likvärdig med den ursprungliga (som baserade sig på mängder).

I mycket av den följande grafteorin kommer bevis och satser att formuleras som om de berör endast grafer, men många av satserna kommer även att vara giltiga för pseudografer, riktade och/eller viktade grafer. Vi kommer dock inte att poängtera detta i de respektive satserna, det lämnas upp till läsaren att vara medveten om distinktionen mellan grafer, pseudografer, riktade och viktade grafer och avgöra vilka typer av grafer som avses. Vi kan konstatera att den mest generella typen av graf är pseudograf och om en sats gäller för pseudografer så gäller den även för de andra typerna av graf. Det är därför lite otillfredsställande att vi inte preciserar begreppet pseudograf noggrannt, men vi avstår ändå från det.

Övningar

Finan: Problem 34.2.

3. Några resultat om grafer

Vi ska ge en sats av Euler om grafer.

Sats: *Eulers Sats.* Låt $G = (V, E)$ vara en graf. Då är summan av alla hörns gradtal lika med 2 gånger antalet kanter. Med andra ord kan vi skriva

$$\sum_{v \in V} deg(v) = 2|E|.$$

Anmärkning: om vi hade valt att beteckna alla noder i V med $v_1, v_2, \ldots, v_n$ skulle vi kunnat skriva $\sum_{v \in V} deg(v)$ som $\sum_{k=1}^{n} deg(v_i)$.

Bevis: Vi kan betrakta summman av alla gradtal på två sätt. Vi kan förstås dels direkt bara summera alla gradtal för alla hörn i grafen och få just talet $\sum_{v \in V} deg(v)$, men vi kan också bilda detta talet genom att summera över alla kanter. Varje kant består ju av precis två noder och varje kant bidrar med talet $2 = 1 + 1$ (1 för båda hörnen som kanten består av) till summan av alla noders gradtal. Så om vi summerar över alla kanter så måste alltså summan av alla gradtal också kunna skrivas som $\sum_{e \in E} 1 + 1$. Vi har alltså

$$\sum_{v \in V} deg(v) = \sum_{e \in E} 1 + 1 = \sum_{e \in E} 2 = 2 \sum_{e \in E} 1 = 2|E|$$

vilket skulle bevisas.

Följdsats: Antalet udda hörn i vilken graf som helst måste vara jämnt.

Bevis: Låt $G(V, E)$ vara en godtycklig graf. Eulers sats ger att ekvationen

$$\sum_{v \in V} deg(v) = \sum_{v \in V, v \ udda} deg(v) + \sum_{v \in V, v \ jämnt} deg(v) = 2|E|$$

gäller för G. Vi delar alltså upp gradtalssumman i summan av alla gradtal av de udda hörnen och summan av alla gradtal av de udda hörnen. Men eftersom summan av alla gradtal av de jämna hörnen måste vara jämn så ger ovanstående uppdelning att även summan av alla gradtal av de udda hörnen måste vara ett jämnt tal, det vill säga talet $\sum_{v \in V, v \ udda} deg(v)$ måste vara jämnt. Eftersom varje term i den summan är udda så måste själva summan innehålla ett *jämnt* antal termer (annars kan den inte bli jämn). Men antalet termer är ju precis lika med antalet udda hörn och detta antal är alltså jämnt vilket skulle bevisas.

Om radar upp alla gradtal som hör till en graf så kan vi alltså vara helt säkra på att antalet gradtal som är udda måste vara jämnt. Vi ska formulera det här på ett alternativt sått och inför därför ett speciellt namn på listan av alla gradtal i en graf.

Definition: Låt $G(V, E)$ vara en godtycklig graf. Då kallas listan på alla gradtal hos noderna i denna graf för grafens *gradtalssekvens*.

Exempel: Gradtalssekvensen för den bipartita grafen ovan är

$$2, 2, 2, 3, 3, 2, 4$$

(gradtalen för hörnen i ordningen a, b, c, d, x, y, z). Vi kan summera dessa gradtal och får då gradtalssumman för grafen som är

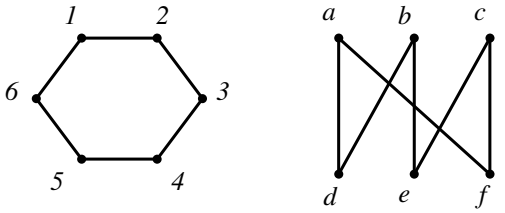
$$\sum_{v \in V} deg(v) = deg(a) + deg(b) + deg(c) + deg(d) + deg(x) + deg(y) + deg(z) = 2 + 2 + 2 + 3 + 3 + 2 + 4 = 18.$$

Enligt Eulers sats är detta tal 2 gånger antalet kanter och om vi räknar kanterna så ser vi att de verkligen är 9 till antalet. Vi observerar också att antalet udda termer i denna summa (som svarar mot antalet udda hörn) är 2 – ett jämnt tal.

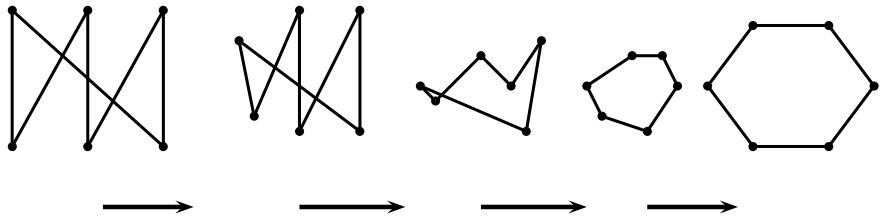
Övningar saknas men teorin kommer att användas i de följande avsnitten så övningar hörande till de andra avsnitten kommer att fungera som övningar till detta avsnitt.

4. Isomorfi

Studera de två följande graferna:



Är det här två olika grafer? Faktiskt inte, de är bara ritade på olika sätt. Om vi formerar de underliggande mängderna av hörn och kanter så kommer vi att se att de bara skiljer sig åt i vilka namn som ges de ingående hörnen. Vi ska snart se detta men vi kan också uppfatta dessa grafers likhet genom en mer bildmässig synvinkel. Två grafer är lika på det här sättet om och endast om vi kan tranformera den ena grafen till den andra genom att tänka oss att de är gjorda av något slags elastiskt material och sedan omformar vi den ena till den andra så att hörn och kanter placeras på nya ställen som överenstämmer med de ställen de finns i den graf vi transformerar till. Själva transformationen är någonting som vi ska se inom oss och det kan illustreras så här:



Vi tänker oss att de här delstegen visar oss att det är möjligt att överföra (eller transformera) den ena grafen till den andra och att detta visar oss att graferna egentligen är en och samma, bara ritade på olika sätt. Vi behöver också tänka oss att transformationen sker inte i delsteg (som i skisserna ovan) utan istället är ett kontinuerligt flöde, som om graferna verkligen vore gjorda av något slag gummi.

När det är möjligt att göra en sådan här tranformation eller omformning och graferna alltså väsentligen är precis samma graf, så säger vi att graferna är *isomorfa*. Det här är förstås grekiska och "iso" betyder "samma" och "morf" betyder "form" så två isomorfa grafer har alltså samma form. (Egentligen kanske det vore bättre att säga att de hade "samma innehåll", men, ja, översättningar fungerar inte alltid så bra.)

Vi ger en skarp definition av isomorfibegreppet och använder oss så av funktioner:

Definition: Givet två grafer $G_1(V_1, E_1), G_2(V_2, E_2)$. Vi säger då att G_1 är *isomorf* med G_2 (eller att G_1 och G_2 är *isomorfa*) och skriver detta $G_1 \cong G_2$ om och endast om det finns en bijektion φ från V_1 till V_2 sådan att:

1. $vw \in E_1 \Rightarrow \varphi(v)\varphi(w) \in E_2$, och
2. $\forall ru \in E_2 : \exists vw \in E_1 : ru = \varphi(v)\varphi(w)$ (vilket bara betyder att $r = \varphi(v)$ och $u = \varphi(w)$).

Vi kallar funktionen φ en *isomorfi* från G_1 till G_2, och vi säger också att $\varphi : G_1 \to G_2$ är en *isomorfi*. (Det är lite oegentligt eftersom φ ju går mellan hörnmängderna.)

För att definitionen ska vara helt konsekvent när den anger att vi kan säga att G_1 och G_2 är isomorfa behöver vi visa att om G_1 är isomorf med G_2 så är G_2 isomorf med G_1, det vill säga vi behöver visa att isomorfirelationen är symmetrisk på mängden av alla grafer. Men det är inte svårt: om φ är en isomorfi som visar att $G_1 \cong G_2$ så fungerar inversen till φ som isomorfi som visar att $G_2 \cong G_1$. Så isomorfirelationen *är* symmetrisk. På samma sätt kan man visa att isomorfirelationen är reflexiv och transitiv. Det betyder att isomorfirelationen är en ekvivalensrelation och vi kan alltså inse att allt som listar ut angående en viss graf gäller för alla grafer som är isomorfa (alltså alla grafer i ekvivalensklassen) så när vi arbetar med grafer får vi ha med oss i åtanke att vi egentligen arbetar med alla grafer som är isomorfa med den som vi har framför oss. Det är ju också det som är själva kärnan i att en graf representerar någonting annat och att genom att arbeta med en graf som representerar någonting så kan slutsatserna om grafen utvidgas till att också gälla det som den representerar.

Det primära sättet att visa att två grafer är isomorfa är att konstruera en isomorfi. Vi gör det med exemplet ovan. Efter en del grubblerier kommer vi fram till att funktionen φ definierad på mängden $\{1, 2, 3, 4, 5, 6\}$ som går till $\{a, b, c, d, e, f\}$ given av $\varphi(1) = a$, $\varphi(2) = f$, $\varphi(3) = c$, $\varphi(4) = e$, $\varphi(5) = b$ och $\varphi(6) = d$ är en isomorfi.

Två isomorfa grafer är alltså väsentligen samma graf: de har samma antal hörn, samma antal kanter, och *hela strukturerna* är överenstämmande. Varje hörn som via isomorfin korresponderar till ett hörn i den andra grafen har samma gradtal som det andra hörnet och så vidare. Det finns inget principiellt behov av att skilja på isomorfa grafer.

Vi kan också betrakta isomorfin som en funktion mellan grafer. Den är ju inte det, isomorfin *är* en funktion mellan hörnmängder men de två kraven isomorfin innebär att vi kan se det som att isomorfin är en avbildning mellan två grafer (en tranformation som vi beskrev i inledningen till avsnittet) och då säger vi att isomrfin *bevarar* strukturerna och egenskaperna på den graf vi stoppar in i isomorfin. Den här egenskapen att en isomorfi bevarar egenskaper kan användas för att visa att två grafer *inte* är isomorfa. Problemställningen är då att vi konfronteras med två olika grafer och ska visa att de inte är isomorfa. Vi kan då visa detta genom att peka ut en egenskap som den ena grafen har som skulle bevaras av en isomorfi, men som den andra grafen inte har. Ett mycket enkelt exempel på detta är antal hörn (eller antal kanter): isomorfa grafer måste ha lika många hörn (eller kanter). Om antal hörn inte överentsstämmer så kan inte graferna vara isomorfa. Vi tar ett par exempel på det här (men inte så uppenbara som de som är baserade på antal hörn).

Exempel: Visa att de två graferna nedan inte är isomorfa.

Vi kan omedelbart se att dessa grafer inte är isomorfa eftersom de har olika gradtalssekvenser. Den till vänster har 3, 2, 2, 1 och den till höger har 2, 2, 2, 2. En isomorfi kan då inte finnas eftersom den ena grafen har hörn av ordningar 1 och 3 och det finns inga sådana hörn i den andra grafen. Om ingen isomorfi finns så kan inte graferna vara isomorfa.

Vi avslutar det här avsnittet genom att ge ett mer detaljerat bevis för att två grafer inte är isomorfa. Det beviset kommer att involvera en mer detaljerad observeration av egenskaperna som de båda graferna har.

Exempel: Visa att de två graferna nedan inte är isomorfa:

Så hur visar vi att dessa två grafer inte är isomorfa? Gradtalssekvenserna hos båda grafer är $4, 2, 2, 1, 1, 1, 1$ så de överenstämmer ... vad finns det för annan egenskap som inte överenstämmer? Vi antar att vi hittat en isomorfi som vi kallar φ. Eftersom båda grafer innehåller exakt ett hörn som har graden 4 så måste isomorfin φ ange att dessa hörn korresponderar mot varandra. Men om vi nu kallar grafen till vänster för G_1 och grafen till höger för G_2 så ser vi att hörnet med grad 4 i G_1 har 4 grannar (förstås!) och att graderna av dessa fyra hörn är $2, 1, 1, 1$. Gradtalen hos de fyra noderna i G_2 som är grannar till hörnet med gradtal 4 i G_2 är emellertid $2, 2, 1, 1$ och det här är en egenskap som måste bevaras av isomorfin: gradtalen hos de fyra grannarna till hörnet med gradtal 4 i G_1 måste överenstämma med gradtalen hos de fyra grannarna till hörnet med grad 4 i G_2 men det är omöjligt eftersom de fyra gradtalen i G_1 är $2, 1, 1, 1$ och de fyra gradtalen i G_2 är $2, 2, 1, 1$. Vi har alltså funnit en egenskap som inte kan bevaras genom en isomorfi, alltså kan det inte finnas någon isomorfi och graferna är alltså inte isomorfa.

Övningar

Finan: Problem 34.8, Problem 34.9.

Bogart-Drysdale-Stein: Exercise 6.1-4.

5. Vägar, cykler, kretsar, ...

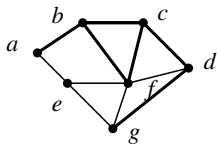
Grafteorin har en massa tekniska termer för att beskriva olika egenskaper som grafer har. Vi ska införa flera sådana begrepp nu för att kunna tala om grafer och deras egenskaper på ett bra sätt.

Definition: Här kommer en lista på viktiga begrepp:

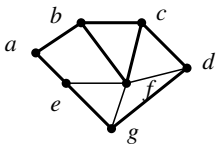
(1) En *promenad* i en graf är en alternerade följd av hörn och kanter som börjar med ett hörn kallat *starthörn* (eller *startpunkt*) och slutar i ett hörn kallat *sluthörn* (eller *sluthörn*). Promenaden har alltså utseendet "starthörn-kant-hörn-kant-...-kant-sluthörn" och varje mellanliggande kant förbinder förstås de två hörn som den förekommer mellan.
(2) *Längden* på en promenad definieras som antal kanter i den.
(3) En promenad är *sluten* om starthörnet sammanfaller med sluthörnet.
(4) En *stig* är en promenad där alla kanter är olika (ingen kant förekommer alltså mer än en gång).
(5) En *väg* är en promenad där alla hörn är olika (inget hörn förekommer alltså mer än en gång).
(6) En sluten stig kallas en *krets*.
(7) En krets där starthörnet sammanfaller med sluthörnet men där inga andra hörn förekommer två gånger kallas en *cykel*.
(8) En *n-cykel* är en cykel med n hörn. Den kallas *jämn* om n är ett jämnt tal respektive *udda* om n är ett udda tal.
(9) En graf kallas *sammanhängande* om det alltid finns en promenad mellan godtyckligt valda par av hörn.

Här har vi alltså en defintion med många begrepp. Vi ska illustrera dessa begrepp genom att ge figurer som förtydligar det mesta, men läsaren uppmuntras att löpande rita egna figurer också.

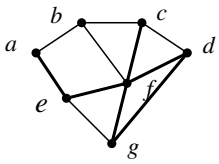
De första fyra figurerna kommer att illustrera en promenad som inte är vare sig en stig eller en väg. Sedan kommer en sluten promenad som inte är vare sig en väg eller stig, sedan kommer en stig som inte är en väg och till sist kommer en väg. Här är dessa figurer:



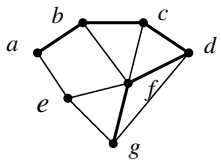
abcfbcdg



abcfbcdgea



aefgdfc



abcdfg

Vi arbetar i samma graf i alla fyra exempel för att göra jämförelser lättare. I alla fyra promenader anger vi dem genom att bara ange de hörn som promenaden passerar genom. (Alla är promenader men alla promenader är inte stigar eller vägar.) Det är överflödigt att ange de mellanliggande kanterna eftersom de här graferna inte är pseudografer som kan ha flera kanter mellan två hörn.

Den första promenaden anges av $abcfbcdg$ och som nämnt ovan är detta en promenad som varken är stig eller väg eftersom både kanter och hörn upprepas. Den andra promenaden är $abcfbcdga$ vilken är densamma som den första men här har vi lagt till innehåll så att promenaden blir sluten.

Sedan har vi $aefgdfc$ som är en öppen stig. Det är en stig eftersom inga kanter upprepas (förekommer två gånger), men hörnet f upprepas så det här är inte en väg.

Slutligen ger vi vägen $abcdfg$ som alltså är en promenad där inga hörn upprepas. Alla hörn förekommer alltså bara en gång och det är precis det som gör den till en väg.

Vi gör en observation här: i en väg upprepas inga hörn. Men det betyder också att ingen *kant* kan upprepas för om en kant upprepas så går vi ju genom den kantens ändpunkter två gånger. Det här betyder att om en promenad är en väg så måste den också vara en stig. Vi kan formulera det lite mer formellt:

Sats: Låt $G(V, E)$ vara en godtycklig graf. För alla promenader $W \subseteq G$ gäller:

$$W \text{ är en väg } \Rightarrow W \text{ är en stig.}$$

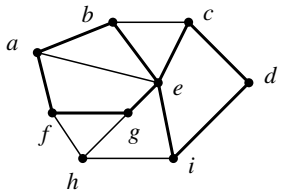
Vi kan förstås formulera den ekvivalenta kontrapositionen som då får lydelsen

$$W \text{ är inte en stig } \Rightarrow W \text{ är inte en väg.}$$

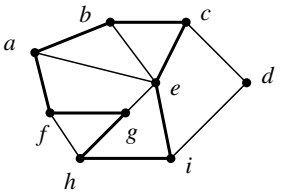
Vi fortsätter nu att ge exempel på en krets (en sluten stig) som inte är en cykel och även en cykel. Det är inte möjligt att ge ett exempel på en "sluten väg" eftersom kravet på en väg är att inga hörn ska upprepas och i en sluten promenad sammanfaller start- och sluthörn. Men en cykel är någonting som vi verkligen ska intressera oss för framöver. På sätt och vis skulle en cykel *nästan* kunna anses vara en "sluten väg", mer precist: en cykel skulle kunna ha varit en väg, det enda som förstör detta är att start- och sluthörn sammanfaller och det är inte tillåtet i en väg.

Det kommer också att stå klart att de begrepp som vi egentligen är intresserade av kommer att vara cykel och väg. De andra begreppen, promenad, stig och i viss mån krets, är mer att betrakta som hjälpbegrepp vars huvudfunktion är att hjälpa oss att införa de mer intressanta begreppen cykel och väg.

Här kommer de utlovade exemplena:



abecdiegfa



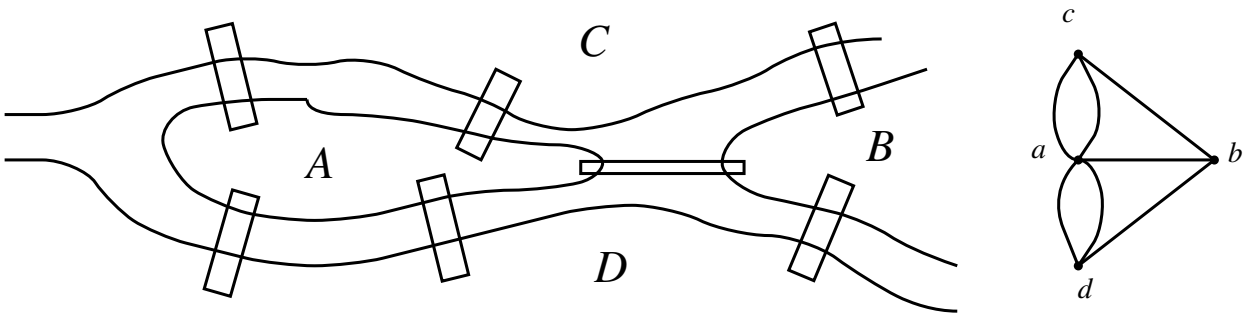
abceihgfa

Det första exemplet ger en krets $abecdiegfa$ som inte är en cykel eftersom till exempel hörnet e upprepas. Lägg dock märke till att inga *kanter* upprepas.

Det andra exemplet $abceihgfa$ är verkligen en cykel precis eftersom inga andra precisely because no vertex (other than the starting vertex a) appears twice.

I dessa exempel har vi inte gett exempel som diskuterat längden eller pariteten (det här med udda och jämna cykler) eftersom dessa begrepp troligen är lättare att förstå. Vi har haft som tonvikt att illustrera just begreppen stig, väg, krets och cykel. Alla grafer vi studerat har varit sammanhängande så det begreppet har dock funnits med i exemplen. (Det är inte så intressant att studera promenader i en icke sammanhängande graf.)

5.1. **Eulerska grafer.** I staden Köningsberg (nuvarande Kaliningrad) fanns sju broar och stadens invånare hade som ett söndagsnöje att försöka ta en promenad från en plats i staden, korsa alla broar precis en gång och komma tillbaka till utgångspunkten. Det här var på 1700-talet så det var förmodligen ett rikemansnöje. Vi ger en principiell karta över staden Köningsberg nedan:



Det här en usel karta, det såg inte ut så här, men vi har idealiserat situationen lite. Vi har idealiserat ännu mer genom att till höger ge en pseudograf som beskriver relationerna mellan landmassorna där den relation vi studerar alltså är att landmassor är eller inte är förbundna med en bro. Det blir en pseudograf eftersom vi har fler än en bro mellan vissa av landmassorna som då alltså illustreras med flera kanter mellan par av hörn i grafen.

Det var aldrig någon som lyckades ta en sådan här promenad och en speciell invånare i staden, som hette Leonhard Euler (1707-1783), insåg att det faktiskt är omöjligt att ta en sådan promenad. Euler själv räknas som en av tidernas mest produktive matematiker, det var han som lade grunden till funktionsbegreppet och talet e (matematikens näst viktigaste tal, efter π) heter e just för att det var Euler som fann det talet. Euler lade också grunden till grafteorin.

Euler tog fasta på egenskapen att i problemställningen skulle promenaden involvera alla broar precis en gång: det betyder att promenaden i pseudografen skulle kunna modelleras som en krets. Men det extra kravet fanns också och det var ju att alla broar skulle besökas. Det här exemplet ligger till grund för begreppet Eulerkrets som vi inför med en definition.

Definition: En Eulerkrets i en graf är en krets som innehåller alla kanter i grafen. En graf kallas *Eulersk* om det finns en Eulerkrets i den. På samma sätt definierar vi en *Eulerstig* som en stig mellan två hörn som innehåller grafens samtliga kanter.

Problemställningen med söndagspromenaden innebar alltså att avgöra om det finns en Eulerkrets i grafen och vi ska ge ett bevis för att det inte finns någon sådan. Pseudografen som modellerar staden Kaliningrad ovan är alltså inte Eulersk.

Bevis: Detta är ett standardmässigt motsägelsebevis. Vi antar att pseudografen ovan har en Eulerkrets. Den startar och slutar då i något av de fyra hörnen a, b, c, d i grafen. Att vi kan gå igenom grafen i en Eulerkrets kräver att vi kan passera varje kant precis en gång. Vad betyder det för hörnens gradtal? Då vi går igenom Eulerkretsen måste vi ankomma till varje hörn (landmassa) och använda en kant (bro) och sedan lämna hörnet (landmassan) genom att använda en annan kant (bro). Det kan hända att vi ankommer samma hörn (landmassa) flera gånger men varje gång måste det ske genom att vi kommer dit via en kant och lämnar via en annan kant. Det betyder att det måste finnas ett *jämnt* antal kanter som ansluter till varje hörn i grafen. Gradtalet förn varje hörn måste alltså vara jämnt om en Eulerkrets existerar. Men om vi studerar gradtalssekvensen för grafen (alltså alla gradtal) så ser vi att den är $5, 3, 3, 3$, det finns alltså inte ett enda jämnt hörn! Vi har på detta sätt nått en motsägelse och vårt antagande om att det skulle finnas en Eulerkrets måste alltså vara felaktigt. Grafen är alltså inte Eulersk och beviset är klart.

Troligtvis var det så här Euler resonerade då han förstörde det här söndagsnöjet. (Förhoppningsvis kanske de som hade tid att ta sådana promenader kunde börja ägna sig mer åt välgörenhet istället.)

I beviset ovan kan vi se en mer generell sats:

Sats: En graf är Eulersk om och endast om alla hörn i grafen är jämna och den är sammanhängande.

Vi ger inte beviset till denna sats, även om halva beviset kan ses i beviset ovan.

Att hitta en Eulerkrets i en graf är handlar verkligen inte bara om nöjen. Det kan vara en väldigt viktig del av ett planeringsarbete när det till exempel gäller underhåll av infrastruktur. När ett lands förvaltande instanser av dess regering ska planera service och reparation av landets vägar är det ytterligt betydelsefullt att vägarna kan nås på ett ekonomiskt sätt. Det är då intressant att hitta saker som Eulerkretsar som kan användas för att besöka varje väg precis en gång. (Att besöka en väg fler gånger än vad som behövs är förstås inte bra ur ekonomisk och miljömässig synvinkel.) Vi kommer att ha mer att säga om sådana här saker när vi studerar algoritmer för att hitta billigaste vägar och andra strukturer i viktade grafer.

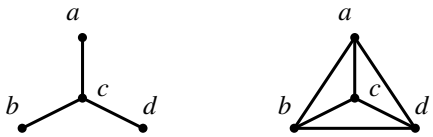
Vi ger till sist en sats som enkelt kan bevisas med hjälp av den föregående satsen om Eulerkretsar:

Theorem: En graf G innehåller en Eulersk stig mellan två hörn $u, v \in G$ om och endast om G är sammanhängande och alla hörn utom u och v i G är jämn.

5.2. **Hamiltoncykler.** Vi kommer inte att studera detta så detaljerat men vi kommer att fokusera på förståelse för vad en Hamiltoncykel är, förmåga att hitta en sådan och viss förmåga att visa att en sådan inte existerar. Vi ger först en definition av detta begrepp.

Definition: En *Hamiltoncykel* i en graf är en cykel som innehåller varje hörn i grafen. En graf som har en Hamiltoncykel kallas *Hamiltonsk*.

Exempel: Betrakta de två givna graferna.



Den till höger är den fullständiga grafen på 4 hörn – K_4 – och den måste därför vara Hamiltonsk. (Är alla fullständiga grafer – K_n – Hamiltonska? Varför? Varför inte?)

Kalla nu grafen till vänster för G. Vi vill visa att G inte är Hamiltonsk, det vill säga att det inte finns någon Hamiltoncykel i G. Det här är en typisk situation då vi vill använda ett motsägelsebevis. Vi antar därför att det finns en Hamiltoncykel H i G. Nu betraktar vi det centrala hörnet c, eftersom det här hörnet är del av cykeln H och hörnen a, b också måste vara del av cykeln då, eftersom det bara kanterna bc och ca existerar så måste ett segment av H innehålla följden acb eller bca. Men d måste också passa in på något sätt och det gå inte eftersom den enda kant som förbinder d är cd och om segmentet acb eller bca med säkerhet ingår i H så kan inte H någonsin nå till d utan att vi gå tillbaka till c vilket skulle innebära att c besökts två gånger vilket då motsäger att H är en Hamiltoncykel. Alltså kan grafen inte vara Hamiltonsk.

Det är föstå enklare för en graf att vara Hamiltonsk ju mer kanter den har och en misstanke vore därför att alla fullständiga grafer K_n är Hamiltonska. En tidigare sats gjorde det möjligt för oss att enkelt avgöra om en graf ï Eulersk, men tyvärr finns det inte något enkelt sätt att avgöra om en graf är Hamiltonsk eller inte. Vi få studera varje fall i detalj och vi kan då vägledas av den här observationen om att det är enklare för en graf att vara Hamiltonsk ju fler kanter den har. Grafen ovan till vänster hade minimalt antal kanter (för att vara sammanhängande) och det visade sig då att den inte var Hamiltonsk. Det enklaste sättet att visa att en graf är Hamiltonsk är förstås om vi lyckas skriva ned en Hamiltoncykel och för grafen ovan (den till höger) kan vi skriva ner Hamiltoncykeln *abcda*.

I digitaltekniken finns två olika tillämpningar av insikter som vi kan få från Hamiltonska grafer. Den första är då vi vill implementera robusta så kallade "asynkrona automater". Ett problem för dessa är att kapplöpning kan uppstå vid tillståndsövergångar men om tillstånden placeras ut som hörn i en n-dimensionell kub och

tillstånden kodas digitalt med egenskaen att en tillstaandsövergång modelleras genom att endast en bit ändras så kan en kapplöpningsfri automat byggas. En annan liknande tillämpning är så kallade *Graykoder* som haft viktiga användningar i felsäkra system.

Övningar

Finan: Problem 34.10.

Bogart-Drysdale-Stein: Exercise 6.3-6, Problem 1 (sidan 298), Problem 2 (sidan 298), Problem 3 (sidan 298), Problem 5 (sidan 298), problem 8 (sidan 298), Problem 12 (sidan 299) – ange bara vilka grafer som har Hamiltoncykler, vi struntar i Diracs och Ores satser,

6. Optimeringsproblem för grafer: Dijkstras algoritm och handelsresandeproblemet

I det här avsnittet och nästa ska vi studera algoritmer som har många mycket viktiga tillämpningar. Den första kallas *Dijkstras algoritm* och varianter av den används varje gång vi ansluter till Internet för att vår användarupplevelse av nätet ska bli så bra som möjligt och så lite energi som möjligt ska spenderas när nätet levererar responser på de tjänster vi begär. Med enklare ord: varje gång vi öpnar en webbläsare kan vi till viss del tacka Edsger Dijkstra (Nederländsk datalog, 1930-2002) för att webbläsaren och webbservern kommunicerar så bra, Dijkstra skapade nämligen en algoritm som hjälper till med detta. Vi ska senare också ge en kort beskrivning av det så kallade handelsresandeproblemet.

6.1. Viktade grafer – igen.

Vi börjar med att ge en alternativ definition av viktad graf. I ett tidigare avsnitt infördes begreppet viktad graf via grannmatrisen, men vi ska göra det genom att utvidga den ursprungliga definitionen av grafbegreppet:

Definition: En *viktad graf* är en graf $G(V, E)$ tillsammans med en reellvärd funktion $w : E \to [0, \infty)$. Om e är en kant i E så kallas det ickenegativa talet $w(e)$ för kanten es *vikt* eller *kostnad*. För en delgraf $G_1 \subseteq G$ definierar vi *kostnaden* för G_1 som summan av alla vikter av alla kanter i G_1. Vi kan också kalla detta tal för *vikten* av G_1.

I grannmatrisen hade vi vikterna som element i själva matrisen som definierade hela grafen, här har vi dock valt ett annat sätt: vi har lagt till en funktion som anger vad vikterna är. Den här definitionen är lite mer generell eftersom vi teoretiskt sett skulle kunna ha en kant med kostnaden 0, men vi arbetar inte ofta med sådana viktade grafer. En 0:a i grannmatrisen betyder ju att det inte finns någon kant så grannmatrisen kan inte definiera grafer med kanter som har kostnad 0. För det behöver vi alltså en mer generell definition och den den ovan är ett exempel på en sådan definition.

Vi ska dock inte analysera definitioner så mycket, det viktiga är att lsaren får en kompetens i att hantera grafer oberoende av hur de definieras.

6.1.1. *En väg betraktad som en delgraf.*

Vi kan anse en väg med en start och en slutpunkt i en graf G som en delgraf av G. En väg är inte exakt definierad som en delgraf, men om vi tar kanterna och hörnen som utgör en väg så är det inga problem med att ändå betrakta en väg som en delgraf. På det viset kan vi tala om *kostnaden* av en väg helt enkelt som kostnaden av vägen betraktad som delgraf.

6.1.2. *Betydelsen av ordet "optimeringsproblem".*

Ordet "optimera" betyder i allmänhet att vi söker ett minimum eller ett maximum av någon storhet ibland med olika bivillkor. Att finna billigaste vägen mellan ett starthörn och ett sluthörn är ett exempel på ett så kallat *optimeringsproblem* där vi alltså försöker hitta ett minimum på en kostnad (som kan definieras som vikten på en delgraf som utgörs av en väg från ett starthörn till ett sluthörn) och villkoren för den minimeringsprocessen kan sägas beskrivas av att vi söker en väg.

Att *optimera* någonting är detsamma som att minimera eller maximera någon variabel (ovan var variabeln *kostnaden* av en väg). Optimeringsproblem kan formuleras på olika sätt och kan ha olika lösningar beroende på hur de formuleras och ordet "optimeringsproblem" betyder alltså att vi har en matematisk problemställning där vi vill söka ett maximum eller ett minimum under vissa givna villkor.

Ett annat exempel på ett matematiskt optimeringsproblem är att söka största och minsta värdet av en deriverbar (reellvärd) funktion definierad på ett slutet och begränsat intervall. Det är välbekant procedur från den matematiska analysen. Optimeringar i den diskreta matematiken handlar däremot om att hantera enskilda

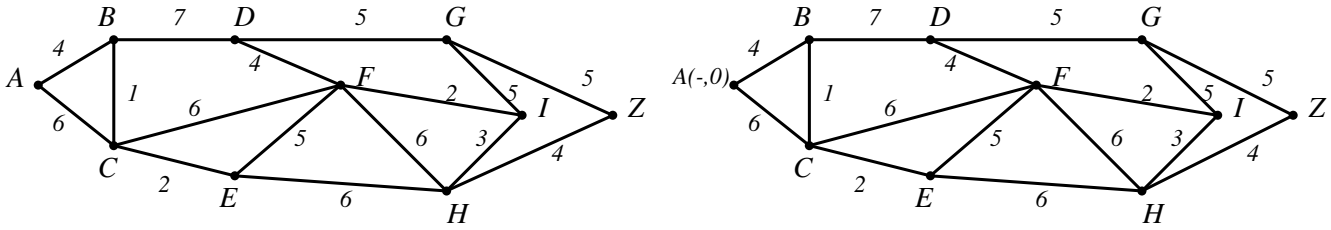
värden och entiteter som vi kan räkna upp och i fallet med ändliga grafer kan vi till och med rita upp en skiss som blir mycket överskådlig.

6.2. **Dijkstras algoritm.** Dijkstras algoritm befattar sig med att finna billigaste vägen som förbinder två hörn, vi bestämmer oss alltså för ett starthörn och ett sluthörn och frågar: "vilken väg med det här starthörnet och det här sluthörnet har lägst kostnad?" Det visar sig att Dijkstras algoritm är ett mycket enkelt sätt att finna en sådan billigaste väg.

Algoritmen går till så att vi sätter så kallade "etiketter" på samtliga noder i grafer. Dessa etiketter kan sedan användas för att finna billigaste vägen från starthörnet till vilket annat hörn i grafen som helst.

Etiketterna sätts ut i en process som hela tiden upprepas, vi tar nya varv i en loop (som i programmering) och i varje varv sätts minst en ny etiketter ut på ett hörn. Dessa etiketter ger oss då mer och mer information om hur billigaste vägar i grafen ser ut. Vi inför Dijkstras algoritm genom att studera ett exempel.

Exempel: Betrakta nedanstående viktade graf och sätt A till starthörn och Z till sluthörn och finn en billigaste väg från starthörnet till sluthörnet (figuren innehåller samma graf i två delsteg av algoritmen – till vänster har vi inte börjat ännu, till höger har vi tagit första steget):



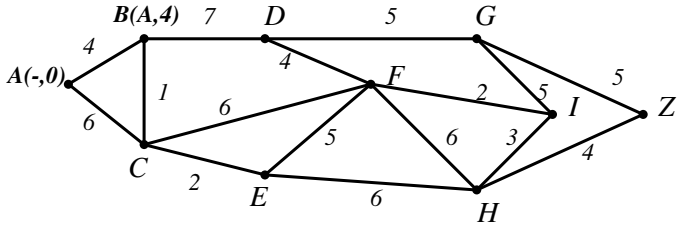
Vi startar alltid Dijkstras algoritm genom att sätta ut etiketten $(-, 0)$ på starthörnet, det indikeras i grafen till höger av att det står $A(-, 0)$. Det ska då tolkas som att "det kostar 0 att ta sig från A till A på billigaste sätt". En etikett som sitter på en nod ger oss precis upplysningen om den lägsta kostnaden att ta sig till den noden frå startnoden.

Vi är nu redo att starta iterationen (upprepningen) i algoritmen som steg för steg ger nya etiketter. Här betraktar vi nu alla hörn som är grannar till hörn med etiketter. Bara A har en etikett än så länge så de nörn som vi betraktar är B och C. Vi ställer oss nu frågan: vilken är billigaste kanten som förbinder B med ett hörn som har en etikett och vilken är billigaste kanten som förbinder C med ett hörn som har en etikett? Eftersom bara A har en etikett i detta steg är svaren på här frågorna att

* kanten AB förbinder B med ett hörn som har en etikett (A själv). Kostnaden av denna kant är 4, respektive att
* kanten AC förbinder C med ett hörn som har en etikett (A själv). Kostnaden av denna kant är 6.

Dijkstras algoritm föreskriver nu att vi väljer ett av dessa svar och skapar en ny etikett. Vi ska välja det svar som har lägst kostnad och det är första svaret. Etiketten som då skapas är $B(A, 4)$. Etiketten har då betydelsen "för att komma från A till B på billigaste sätt ska vi komma från A och det kostar 4". (Första etiketten som sattes ut på A kanske därför skulle haft utseendet $A(A, 0)$.)

Observera att vi i det här steget också har på förslag att hörnet C ska få etiketten $C(A, 6)$. Vi kan då betrakta $B(A, 4)$ och $C(A, 6)$ som *kandidater* till nästa etikett och $B(A, 4)$ vinner eftersom den har lägst kostnad. Efter vi satt ut den vinnande etiketten har vi nedanstående utseende.



För att vi lättare ska få en känsla för algoritmens väsen eller grundidé inför vi en liknelse. Om vi tänker oss att den viktade grafen är som ett vatten kärl där hörnen kan fyllas av vatten och att vattnet kan flyta genom kanterna så har vattnet en förmåga att alltid hitta kortaste vägen genom att vattenytan alltid är plan. Om vi då tänker oss att vatten kommer in frå starthörnet och fyller hela grafen så har nu vattenytan stigit så att

hörnet A stå under vatten men också hörnet B. Vi ska också vara mer formella och införa beteckningen U för mängden av hörn som har etiketter (alltså hörnen som "står under vatten"). I första steget när bara A hade etiketten $A(-,0)$ var $U = \{A\}$ men nu när andra steget är taget och B fått etiketten $B(A,4)$ är $U = \{A, B\}$.

Innan vi kör vidare ska vi göra en fullständig formulering av Dijkstras algoritm och sedan ska vi fortsätta att köra algoritmen så att hela grafen förses med etiketter. Det kan hända att det är lättas att läsa igenom hela exemplet före du läser den formella beskrivningen av algoritmen.

Formell beskrivning av Dijkstras algoritm. (Vi antar att starthörnet betecknas A.)

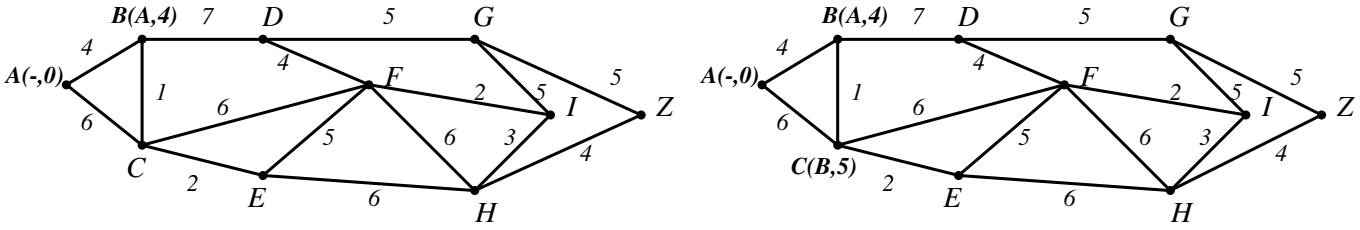
Steg 0. Sätt ut etiketten $A(-,0)$ och beteckna löpande med U mängden av hörn som har etiketter utsatta. (Efter steg 0 är alltså $U = \{A\}$.)

Steg 1. Beteckna med V mängden av de hörn i grafen som är grannar med hörn som har etiketter, alltså hörnen i U. För varje hörn $v \in V$ bilda en kandidatetikett på följande sätt: finn det hörn $u \in U$ som minimerar $w(uv)+d_u$ där d_u är vikten i etiketten på hörnet u. Kandidatetiketten är då $v(u, w(uv)+d_u)$. (I exemplet ovan uppstod kandidaterna $B(A,4)$ och $C(A,6)$.)

Steg 2. Välj den kandidat som har minst kostnad och utöka mängden U genom att införa den aktuella etiketten på det aktuella hörnet. Om alla hörn har etiketter är vi klara, annars upprepa steg 1 och 2 tills alla hörn har etiketter. (Det går att avbryta så fort sluthörnet har fått en etikett.)

Steg 3. Den billigaste vägen från starthörnet till sluthörnet kan nu tas fram med hjälp av etiketterna. (Vi illustrerar det sist i det givna exemplet.)

Så vi illustrerar nu hur det här fungerar när vi tar steget från $U = \{A, B\}$ i vårt exempel. Till vänster i figuren nedan ser vi situationen $U = \{A, B\}$ och vi ska nu lägga till nästa etikett.



Med $U = \{A, B\}$ får vi mängder av grannhörn till $V = \{C, D\}$ och de olika möjliga valen av (u, v) i detta steg blir då

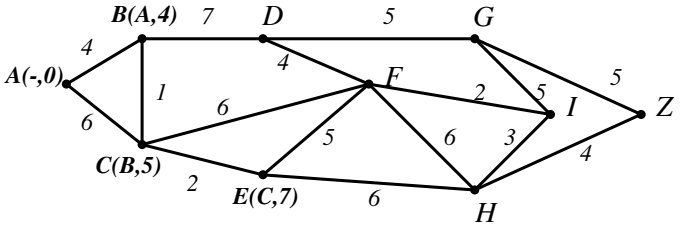
(1) $(u, v) = (A, C) \Rightarrow d = 6$ $\quad (d = 6 + 0 = w(AC) + 0)$
(2) $(u, v) = (B, C) \Rightarrow d = 5$ $\quad (d = 4 + 1 = w(AB) + 1)$
(3) $(u, v) = (B, D) \Rightarrow d = 11$ $\quad (d = 7 + 4 = w(AB) + 4)$

Här betecknar d vikten på den kandidatetikett som uppstår och av denna lista skapar vi de tre kandidatetiketterna $C(A,6)$, $C(B,5)$ och $D(B,11)$. Av dessa representerar $C(B,5)$ den minsta kostnaden varför vi sätter etiketten $(B,5)$ på hörnet C. Mängden U är nu utökad till $\{A, B, C\}$ och vi ser etiketten på plats till höger i figuren.

Vi ser i det här läget att algoritmen insett att det är billigare att gå från A till B *via* B, det kostar alltså 5 att komma till C om vi kommer frå B, det är betydelsen av etiketten som just placerats på C och i precis den här situationen så kan vi alltså inse varför Dijkstras algoritm fungerar.

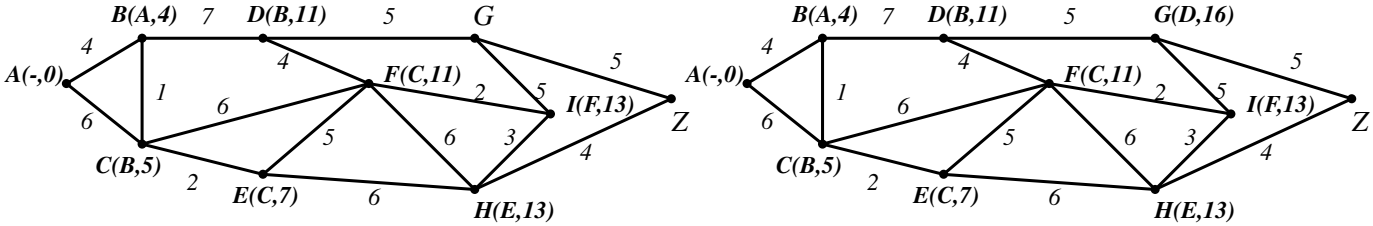
Men vi betonar också den tidigare liknelsen med en vattennivå som hela tiden stiger. Vattnet har nu alltså stigit så att A, B och C stå under vatten och vattnet har flutit genom vägen ABC och inte direkt från A till C eftersom det innebär en större kostnad.

Vi fortsätter processen med $U = \{A, B, C\}$. Då blir $V = \{D, F, E\}$. Möjligheterna att välja (u, v) blir $(B, D) \Rightarrow d = 11$, $(C, F) \Rightarrow d = 11$ och $(C, E) \Rightarrow d = 7$ och vi få alltså de tre kandidaterna $D(B, 11)$, $F(C, 11)$ och $E(C, 7)$, där $(u, v) = (C, E)$ ger den minsta kostnaden $d = 7$ (som gav upphov till kandidaten $E(C, 7)$) så nästa etikett blir $(C, 7)$ på hörnet E. Detta representeras bildmässigt till vänster i nedanstående figur.



Och nu kommer vi alltså till $U = \{A, B, C, E\}$ och $V = \{D, F, H\}$. Valen av (u, v) är $(B, D) \Rightarrow d = 11$, $(C, F) \Rightarrow d = 11$, $(E, F) \Rightarrow d = 12$ och $(E, H) \Rightarrow d = 13$. Här har vi en intressant situation. Vi har två kandidater som ger samma minimala kostnad i sina respektive etikettförslag. Om dessa kandidater till (u, v) (det vill säga (B, D) och (C, F)) med tillhörande etiketter, svarade mot samma un eller samma v så skulle vi kunna välja att använda *något* av de båda förslagen, men eftersom dessa båda förslag skiljer sig åt både i förstakomponenterna (alltså u:na – $B \neq C$) *och* i andrakomponenterna (alltså v:na – $D \neq F$) så kan vi använda *båda* etikettförslagen. Vi sätter alltså dit etiketterna $D(B, 11)$ och $F(C, 11)$ samtidigt. Detta kan ses som att vattenytan har stigit och når nu D och F samtidigt, de ligger alltså på samma minimala avstånd från A. Vi kan också inse att vi skulle kunnat välja bara en av etiketterna men i nästa iteration av algoritmen så skulle då den etikett vi inte valt dykt upp så båda etiketterna kan sättas på samma gång.

Ja, så nu har vi $U = \{A, B, C, D, E, F\}$ och det ger i detta steg $V = \{G, I, H\}$. Valen av (u, v) blir nu $(D, G) \Rightarrow d = 16$, $(F, I) \Rightarrow d = 13$, $(F, H) \Rightarrow d = 17$ och $(E, H) \Rightarrow d = 13$ och vi får återigen två etiketter på en gång: $I(F, 13)$ och $H(E, 13)$. Detta illustreras i figuren nedan till vänster.



Vi börjar nu närma oss slutet, vattennivån har stigit så att den täcker alla hörnen A, B, C, D, E, F, I, H och endast G, Z är kvar. Vi kan nu nå Z (sluthörnet) och detta uttrycker sig i att Z är en medlem i V. Vi har alltså $U = \{A, B, C, D, E, F, I, H\}$ och $V = \{G, Z\}$. Valen av (u, v) ges nu av $(D, G) \Rightarrow d = 16$, $(I, G) \Rightarrow d = 18$ och $(H, Z) \Rightarrow d = 17$ vilket ger oss etiketten $G(D, 16)$ som illustreras i figure ovan till höger.

Till sist är det mycket enkelt att hitta den sista etiketten för Z och den blir $(H, 17)$.

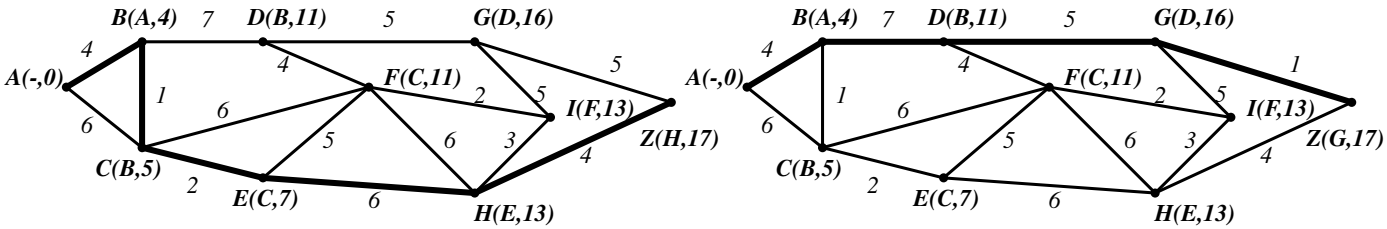
Vi ska nu använda etiketterna för att ta fram billigaste vägen från A till Z. Vi gör detta genom att utgå från sluthörnet Z och läsa av i etiketten varifrån vi har kommit för att ha gått den kortaste vägen. Om vi läser baklänges på detta sätt får vi kortaste vägen från A till Z given av

$$A - B - C - E - H - Z$$

och den har kostnaden 17 (som ju också anges av etiketten på Z).

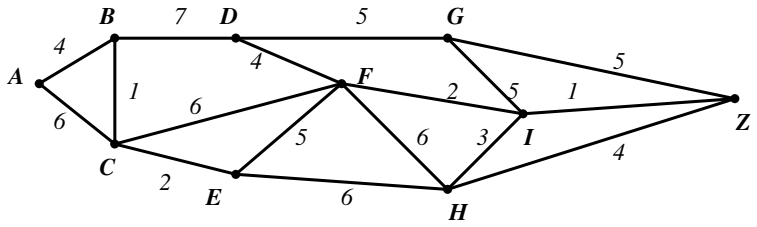
Väl att märka i det här skedet är att etiketterna som sitter på alla hörn i grafen ger oss möjlighet att hitta kortaste vägen från A till *vilket annat hörn som helst i grafen.* Kortaste vägen från A till F ger till exempel av $A - B - C - F$ och den har kostnaden 11.

Vi illustrerar kortaste vägen från A till Z till vänster i nedanstående figur.



Till höger ovan har vi en annan situation där vi ändrat vikten av kanten GZ till 1. Det får effekten att en helt ny billigaste väg från A till Z uppkommer. Vi har i själva verket två alternativa kortaste vägar här men poängen är att etiketterna så att säga innehåller kunskap om grafens struktur och att en liten ändring i

förutsättningarna (vikten på GZ sänktes) gav upphov till en ganska stor ändring i resultatet. En annan liten ändring som också ger en stor ändring i resultatet är om vi lägger till kanten IZ med vikten 1. Detta ger upphov till en helt annan billigaste väg från A till Z. Läsaren uppmanas köra algoritmen och se vad som händer. Vi illustrerar detta nedan.



6.3. **Handelsresandeproblemet.** Dijkstras algoritm som vi gått igenom ovan löser det enkla optimeringsproblemet att hitta en billigaste väg med ett givet starthörn och ett givet sluthörn. (Och som vi anmärkte behöver inte sluthörnet bestämmas.)

Om vi dock skulle lägga till villkoret i problemställningen med att hitta billigaste vägen mellan ett starthörn och ett sluthörn att vi också måste besöka *alla* hörn i grafen och återkomma till utgångspunkten så att vi har en Hamiltoncykel så får vi genast ett *mycket* svårt problem. Det kallas för *handelsresandeproblemet* för det modellerar en tänkt situation med en handelsresande som vill besöka olika orter och resa mellan dem på det mest ekonomiska sättet. Det är som sagt ett väldigt svårt problem och vi kan absolut inte ge en lösning på det men vi kan förstås tänka oss att problemet kan bli lättare vi lägger till bivillkor som att vissa delar av grafens hörn beshöver besökas i en viss ordning. Det skulle kunna vara en modell av verkligheten där en handelsresande, för att överhuvudtaget kunna bedriva sin verksamhet, ï tvungen att besöka vissa orter av central betydelse, som till exempel ett lager där varor förvaras och liknande. Vi ska dock inte fördjupa oss i detta men nämner handelsresandeproblemet för läsarens allmänbildnings skull.

Övningar på optimeringsproblem kommer i de blandade problemen.

7. Träd och minimala uppspännande träd

Vi ska inför mycket viktig typ av grafer: så kallade *träd*. Grafer som är träd har två egenskaper: de är sammanhängande och mellan varje par av hörn finns exakt en väg. Det är så pass viktigt att vi ger en formell definition.

Definition: Låt $G(V, E)$ vara en graf. Då kallas $G(V, E)$ ett *träd* om och endast om G har följande två egenskaper:

(1) G är sammanhängande.
(2) G för varje par av hörn v, w i G gäller att det finns exakt en väg med v som starthörn och w som sluthörn.

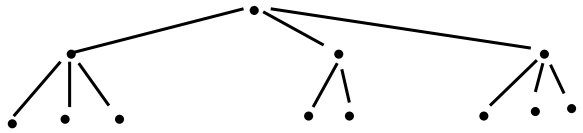
Ett hörn av ordning 1 i ett träd kallas ett *löv*.

Vi inser lätt att följande sats gäller:

Sats: Låt G vara en sammanhängande graf. Då är G ett träd om och endast om det inte finns några cykler i G.

Satsen visar att vi också skulle ha kunnat ta det här som definition på ett träd, alltså en graf sägs vara ett träd om och endast om grafen är *sammanhängande och inte har några cykler*.

Grafen som vi såg i början av detta kapitel som hade följande utseende:



är ett exempel på ett träd. De hörn som ritats ut längst ner i den här bildmässiga representationen har som vi ser ordning 1 och är därför de så kallade löven. Man kan på motsvarande sätt även kalla ett visst av trädets hörn för en *rot* och med det vill vi införa ett slags utgångspunkt i trädet. Om trädet modellerar en filstruktur

där hörnen symboliserar kataloger och löven är filer som ligger i katalogerna så kan man utse en katalog till den som ligger högst upp och det hörnet benämns då trädets *rot*. I datateknik används också ordet "rotkatalog" eller bara rätt och slätt ordet "roten" till en viss enhet (hårddisk eller liknande). Viktigt att lägga märke till är att vi kan välja vilket hörn som ska anses vara rot (och då presenterar vi i allmänhet också trädet på ett annat sätt genom att rita roten överst).

Om vi tänker på vad som krävs för att att en graf ska vara ett träd så har vi två krav: 1. den ska vara sammanhängande. 2. det ska bara finnas *en* väg mellan varje par av hörn. En graf kan inte vara sammanhängande om den har för få kanter. Det första kravet på att trädet ska vara en sammanhängande graf innebär alltså att grafen har tillräckling många kanter. Men det andra kravet lägger en gräns på hur mycket kanter vi kan ha. Om vi har för många kanter i en graf, ja då uppstå cykler, så dessa två krav verkar skapa ett slags dragkamp. Och fraktiskt skapas en balans mellan antal hörn och kanter av dessa två krav. Vi har följande viktiga sats om träd:

Sats: Låt $G(V, E)$ vara ett träd. Då gäller $|V| - 1 = |E|$.

Antal hörn är alltså precis ett mer än antalet kanter. Vi ska ge ett bevis för det här men vi ska först visa att ett träd alltid måste ha minst ett löv. Vi ger det som ett lemma (alltså en hjälpsats) innan vi visar satsen om antal hörn och kanter i ett träd.

Lemma: Låt grafen $G(V, E)$ vara ett träd med mer än ett hörn men med ett ändligt antal hörn. Då har G minst ett löv.

(I lemmat lägger vi till förutsättningen att vi har mer än ett hörn för att utelämna det löjliga trädet som bara består av ett hörn och inga kanter – en enda isolerat hörn alltså, inte intressant.)

Bevis av lemmat: Antag motsatsen till det som ska visas, det vill säga antag att trädet inte har något löv, det vill säga hörn av grad 1. Eftersom trädet är en sammanhängande graf så kan inte några hörn vara isolerade (det vill säga ha grad 0). Grafens *samtliga* hörn måste alltså vara av ordning 2 eller högre. Vi kan nu välja ett hörn som vi kallar e_0 i grafen och bilda en följd av hörn, $e_1, e_2, \ldots$ genom att e_1 väljs som ett hörn som är granne till e_0, eftersom e_1 har ordning ≥ 2 så kan e_2 väljas som en granne till e_1 som *inte* är e_0 och så vidare, en följd av hörn bildas $\ldots, e_{n-1}, e_n, e_{n+1}, \ldots$ där e_{n-1} är granne till e_n som är granne till e_{n+1} och så vidare i all oändlighet – notera att vi kan aldrig upprepa hörn i den här följden, då skulle vi skapa en cykel och ett träd har ju inga cykler. Men detta motsäger att trädet ska innehålla ett ändligt antal hörn totalt. Antagandet om att det inte finns något hörn av grad 1 (ett löv) måste alltså vara felaktigt, alltså finns ett löv och beviset är klart.

Som vi nämnde i början av kapitlet studerar vi bara ändliga grafer (som alltså har ett ändligt antal hörn) men det är egentligen bara i den här satsen (eller lemmat) som vi använder det så vi formulerar i lemmats förutsättningar kravet att antalet hörn är ändligt.

Nu kan vi bevisa satsen om antal hörn och antal kanter i ett träd:

Bevis: Vi använder matematisk induktion över antalet hörn i trädet. Inför alltså predikatet

$$A(n) \Leftrightarrow \text{ ett träd med } n \text{ hörn måste ha } n-1 \text{ kanter.}$$

Vi ska nu visa $\forall n \in \mathbb{N} : A(n)$.

Steg 1. Kolla att $A(1)$ gäller. Ett träd med bara ett hörn (det löjliga trädet!) kan inte ha några kanter, antal kanter måste alltså vara 0 och det är ju precis ett mindre än antal hörn så $A(1)$ gäller.

Steg 2. Visa nu att $A(p) \Rightarrow A(p+1)$ för alla $p \geq 1$. Som vanligt antar vi därför att $A(p)$ är sann, det vill säga om ett träd har p hörn så måste det innehålla $p-1$ kanter. Vi ska nu visa att $A(p+1)$ är sann så väljer ett godtyckligt träd med $p+1$ hörn, vi betecknar det trädet med $T(V, E)$, där alltså V är mängden av hörn och E är mängden av kanter i trädet. Enligt lemmat ovan så har detta träd ett löv, vi kallar det lövet e. Eftersom e är ett löv så ansluts e till T med en kant som vi benämner v. Om vi nu tar bort e och v från T så uppstå en ny graf som vi kan kalla $G(V', E')$, där alltså $V' = V - \{v\}$ och $E' = E - \{e\}$. Vi har tagit bort lövet och den enda kant som förband lövet med det övriga trädet. Den nya graf som uppstå har alltså $|V'| = |V| - 1 = p + 1 - 1 = p$ respektive $|E'| = |E| - 1$. Men eftersom vi tog bort ett löv från ett träd och den tillhörande kanten så har vi skapat en graf som också måste vara ett träd: den nya grafen måste vara sammanhängande för om den inte vore sammanhängande så hade inte trädet vi utgått från varit sammanhängande och det kan av

samma anledning inte finnas cykler. Det betyder att (V', e') är ett trd med p hörn och det uppfyller alltså förutsättningara i induktionsantagandet, vi kan alltså här använda att

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

men detta ger då med sambanden $|V'| = |V| - 1 = p + 1 - 1 = p$ respektive $|E'| = |E| - 1$ att

$$|V| - 1 = |E| - 1 + 1 \Leftrightarrow |V| = |E| + 1$$

vilket precis uttrycker att antal hörn i G är precis ett mer än antalet kanter. Eftersom G var ett godtyckligt valt träd med $p + 1$ hörn måste alltså $A(p + 1)$ vara sann och vi har visat implikationen $A(p) \Rightarrow A(p + 1)$. Steg 2 i induktionsbeviset är klart.

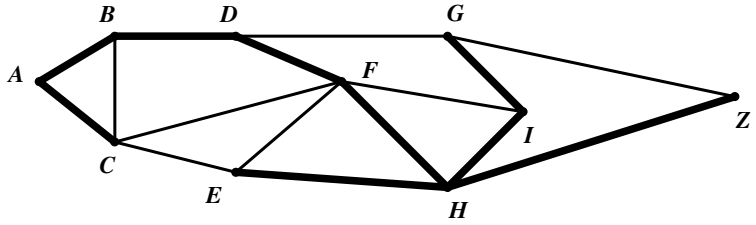
Steg 3. Steg 1 och 2 och induktionsaxiomet fullbordar beviset och vi har alltså alltid att antalet hörn i ett träd är precis ett mer än antalet kanter.

Bevis som involverar grafer har ganska mycket text i sig, och det hänger på att många av begreppen definieras med det naturliga språket så att skriva ner bevis som gäller grafer kräver alltså att vi formulerar oss noggrannt.

7.1. **Uppspännande träd, Prims och Kruskals algoritmer.** Vi vill nu studera en speciell typ av viktiga optimeringsproblem som involverar träd och för den skull ska vi först införa en speciell typ av träd som kan bilda ett slags skelett till en graf. Det kallas "uppspännande träd" och vi ger en definition av det här.

Definition: Låt $G(V, E)$ vara en sammanhängande graf. En delgraf $T \subseteq G$ med egenskaperna att T innehåller alla hörn som finns i G och att T också är ett träd kallas ett *uppspännande träd* för G.

Ett uppspännande träd anges ofta tillsammans med den graf som det spänner upp, om vi studerar grafen som vi hade förut så kan ett uppspännande träd för den grafen anges genom att vi ritar de kanter som ingår i det uppspännande trädet tjockare än de kanter som inte ska ingå. Det kan se ut så här:



Med det här vill vi alltså symbolisera det uppspännande trädet

$$(\{A, B, C, D, E, F, G, H, I, Z\}, \{AB, AC, BD, DF, EH, FH, GI, HI, HZ\})$$

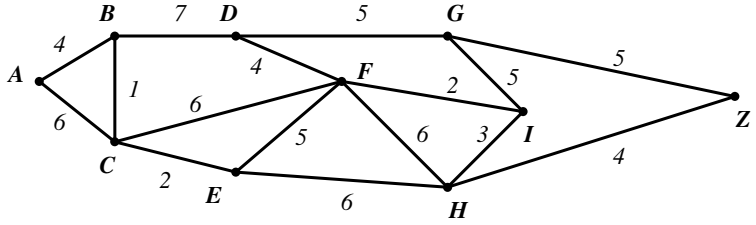
som alltså är ett uppspännande träd till den givna grafen. Vi har i detta träd med samtliga 10 hörn från den graf som vi ska spänna upp men då bara 9 av dessa kanter.

Ett uppspännande träd skulle kunna modellera ett strömförsörjningsnät mellan orter i ett landsområde eller komponenter i en maskin. Möjligheterna är, som så ofta med grafer, oerhört vittomspännande.

Det optimeringsproblem som vi nu ska studera är att finna ett så kallat *minimalt* uppspännande träd: vi vill alltså hitta ett uppspännade tråd i en viktad graf med total kostnad varandes så liten som möjligt.

Vi ska studera detta problem genom att arbeta med samma graf som exempel, vi utgå alltså från den viktade grafen som vi förut använde för att illustrera Dijkstras algoritm.

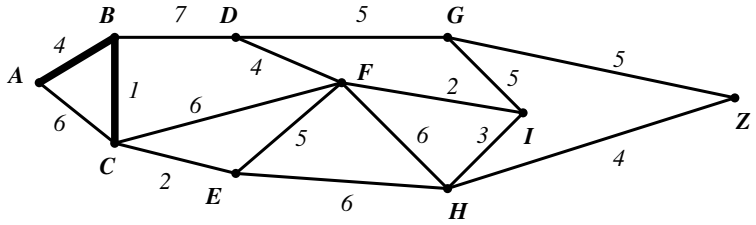
Exempel: Finn ett minimalt uppspännande träd i nedanstående viktade graf.



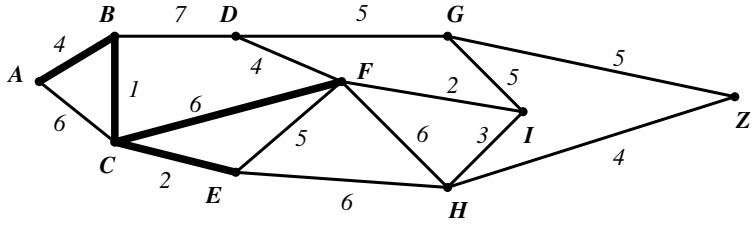
Vi ska göra detta genom att steg för steg illustrera det som kallas *Prims algoritm*. Det kommer att vara ännu enklare än Dijkstras algoritm och vi beskriver algoritmen genom att köra den på den givna grafen.

Alla hörn ska ingå i ett minimalt uppspännande träd så vi kan inrikta oss på ett hörn, till exempel A. Någon av kanterna som ansluter till A (alltså AB med vikt 4 eller AC med vikt 6) måste ingå i trädet så och eftersom trädet ska vara minimalt väljer vi kanten AB och vi har nu början till ett minimalt uppspännande träd. Vi fortsätter och addera kanter och hörn och får ett större och större träd som till slut innehåller alla hörn och en mängd kanter som ger en minimal kostnad.

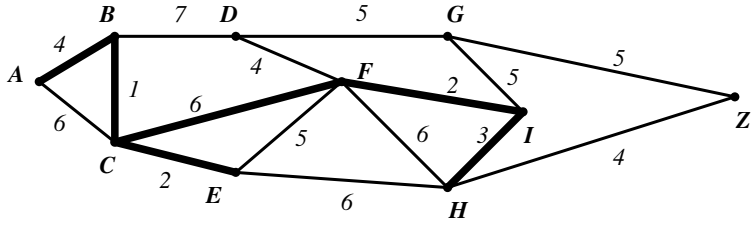
Vi har alltså hörnen A och B tillsammans med kanten AB och fortsätter att bygga på detta. Vi studerar alla intilliggande kanter och dessa utgörs av AC med vikten 6, BC med vikten 1 och BD med vikten 7. På samma sätt som förut väljer vi kanten BC eftersom den har lägst vikt och trädet har nu hörnen A, B, C och kanterna AB, BC. Vi illustrerar det hela i grafen nedan.



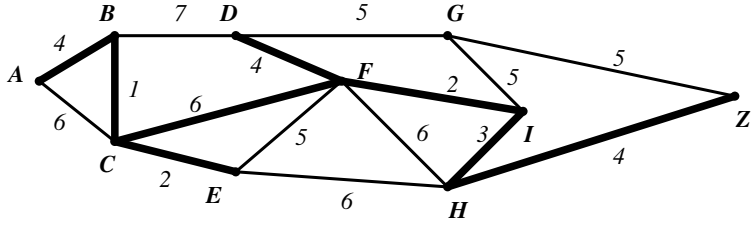
I nästa steg behöver vi inte överväga om vi ska lägga till kanten AC eftersom den går mellan två hörn som redan finns i trädet. Vi ska bara överväga kanter som leder ut från trädet till hörn som inte är i trädet. När vi nu har trädet med hörn A, B, C så är det kanterna BD (vikt 7), CF (vikt 6) respektive CE (vikt 2). Vi väljer den billigaste som är kanten CE och utökar trädet med hörnet E och kanten CE. Trädet har i detta steg hörnen A, B, C, E och kanterna AB, BC och CE. Lägg märke till hur trädet hela tiden har en kant mindre än antalet hörn och att i varje steg så utökas trädet med en kant från grafen som ökar trädets vikt minimalt. I nästa steg har vi att välja på kanterna BD (vikt 7), CF (vikt 6), EF (vikt 5) respektive EH (vikt 6) och vi väljer förstås kanten EF och vi trädet vi få illustreras nedan:



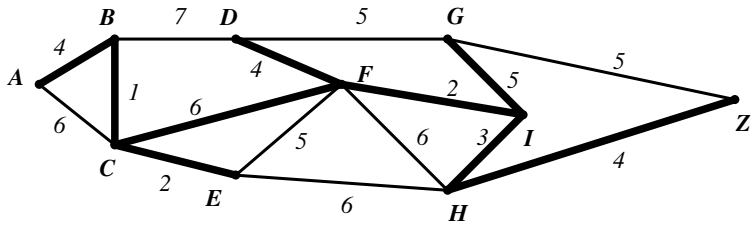
Och trädet ges nu av hörnen A, B, C, E, F med kanterna AB, BC, CF, CE. Nästa två kanter som läggs till blir först FI (vikt 2) och sedan HI (vikt 3) och trädet har då följande utseende:



och i det här läget har vi kommit till en valsituation: vi kan antingen addera kanten HZ eller kanten DF, de har båda vikten 4. Men det spelar ingen roll, om vi lägger till den ena så kommer vi i nästa steg att garanterat lägga till den andra så vi lägger till båda två och får trädet som illustreras nedan:



I sista steget har vi tre alternativ, vi kan addera vilken som helst av kanterna DG, IG eller GZ, alla har samma vikt och vi väljer IG och får det slutliga minimala uppspännande trädet givet nedan:



Det minimala uppspännande trädet ges alltså av

$$(\{A, B, C, D, E, F, G, H, I, Z\}, \{AB, BC, CE, CF, FD, FI, IH, IG, HZ\})$$

och det har vikten $4 + 1 + 2 + 6 + 4 + 2 + 3 + 5 + 4 = 31$.

Prims algoritm utgår alltså från grafens kanter och bygger successivt upp ett träd som till slut innehäller grafens alla hörn. Trädet byggs upp på ett sådant sätt att det bli minimalt och till sist kommer det att spänna upp hela grafen.

Det finns ett alternativ till Prims algoritm som brukar benämnas Kruskals algoritm, där utgår vi istället från kanterna och adderar dem i storleksordning och undviker att addera kanter som ger upphov till cykler. Om vi skulle kört Kruskals algoritm på ovanstående graf så kunde kanter adderats i följande ordning:

$$BC, CE, FI, IH, AB, DF, HZ, IG, CF.$$

Det finns ofta flera olika minimala uppspännande träd men de har föstås alla samma minimala vikt.

Övningar

Finan: Exempel 35.1.

Bogart-Drysdale-Stein: Exercise 6.1-6, Exercise 6.1-7, Exercise 6.1-9, Exercise 6.1-10, Problem 1 (sidan 274), Problem 12 (sidan 274), Problem 13 (sidan 274), Problem 15 (sidan 275), Problem 16 (sidan 275), Exercise 6.2-1, Problem 1 (sidan 285), Problem 2 (sidan 285)

Blandade övningar

Samtliga blandade övningar är hämtade från gamla tentor så de anges inte här så ni kan gå direkt på högen med gamla tentor nu.