Mittuniversitetet
MID SWEDEN UNIVERSITY

Block 6
Algebra och Diskret Matematik A

BLOCK INNEHÅLL

Referenser

Inledning

1. Grafer
   - Introduktion och terminologi
2. Stigar och cykler
   - Eulercykler
   - Handskakningslemmat
   - Hamiltoncyckler och handelseresandeproblemet
   - Dijkstra's algoritm för kortaste stig
3. Grannmatrisen och incidensmatrisen för en graf
4. Grafisomorfier
5. Träd
   - Introduktion
   - Definitioner och terminologi
   - Uppspännande träd
   - Bredd före djup-sökning
   - Djup före bredd-sökning
6. Minsta uppspännande träd
   - Prims algoritm
   - Kruskals algoritm
7. Övningsuppgifter

Grafteori

Referenser

[J] Edition 5: Nedanstående text + [J] 6.1 - 6.6, 7.1 - 7.4
[J] Edition 4: Nedanstående text + [J] (6.1) - (6.6), (7.1) - (7.4)
[J] Edition 6: Nedanstående text + [J] [8.1] - [8.6], [9.1] - [9.4]

Bemärk, referenser i texten utan parenteser är till [J]ohnsonbaugh Edition 5, referenser i ()-parenteser är till [J] Edition 4, och referenser i []-parenteser är till [J] Edition 6.

Inledning

Denna del av den diskreta matematiken är för mig den mest spännande. Många av problemen är väldigt lätta att förstå, men lösningarna kan vara så svåra att det nästan krävs en doktorsexamen för att ens kunna inse riktigheten i dem! Flera problem som vi möter varje dag kan modelleras med grafteori. De följande är några exempel.

- Hur kan en utslagsturnering i tennis arrangeras så att precis fyra spelare återstår till semifinalerna?
- Hur ska kablarna dras så att kostnaden för bredband till alla i Sverige blir så låg som möjligt?
- Hur reser man billigast från Söråker till Birmingham i England?
- Hur kan bensenmolekyler vara sammansatta?
- Hur ska scheman läggas så att konflikter undviks, d.v.s. så att varje student bara har en lektion i taget?
- Hur är det möjligt att ge var och en av tio frivilliga ett av tio arbeten de är kvalificerade för?
- Hur många färger behövs för att färga en karta så att länder med gemensam gräns ej färgas i samma färg?
- Hur bör man färdas om man vill besöka alla vinslotten i Frankrike med så kort resväg som möjligt?
- Bredd-först och djup-först sökmetoderna är baserade på träd, som är en speciell slags grafer.

1. Grafer

Grafteori är inte konsekvent när det gäller terminologi, dvs. olika böcker använder olika ord för samma sak. På engelska finns det åtminstone två uppsättningar med terminologi, och några blandar dem t.o.m. Byter man bok om grafteori är det väldigt viktigt att kontrollera vilka definitioner som används i den nya boken. Som exempel, en **stig** (path) som uppfyller definition 6.2.1 (6.2.1) [8.2.1] i [J] tillåter att både hörn och kanter upprepas, medan man i annan litteratur varken tillåter upprepade hörn eller upprepade kanter i en stig. Var uppmärksam på detta!

Notera att en graf består av **hörn** (vertices) och **kanter** (edges) och *inte* punkter och linjer. En möjlig förklaring till varför vi inte säger punkter och linjer är den följande. En linje går igenom flera punkter men en kant har bara hörn vid sina ändpunkter. En linje skapar en idé om en given form, d.v.s. en rak linje, medan en kant kan vara krokig som en slalombacke!

Vid visst datorarbete kan det ibland vara fördelaktigt att studera grafer *utan* hörn men vi ska i denna kurs studera grafer med åtminstone ett hörn.

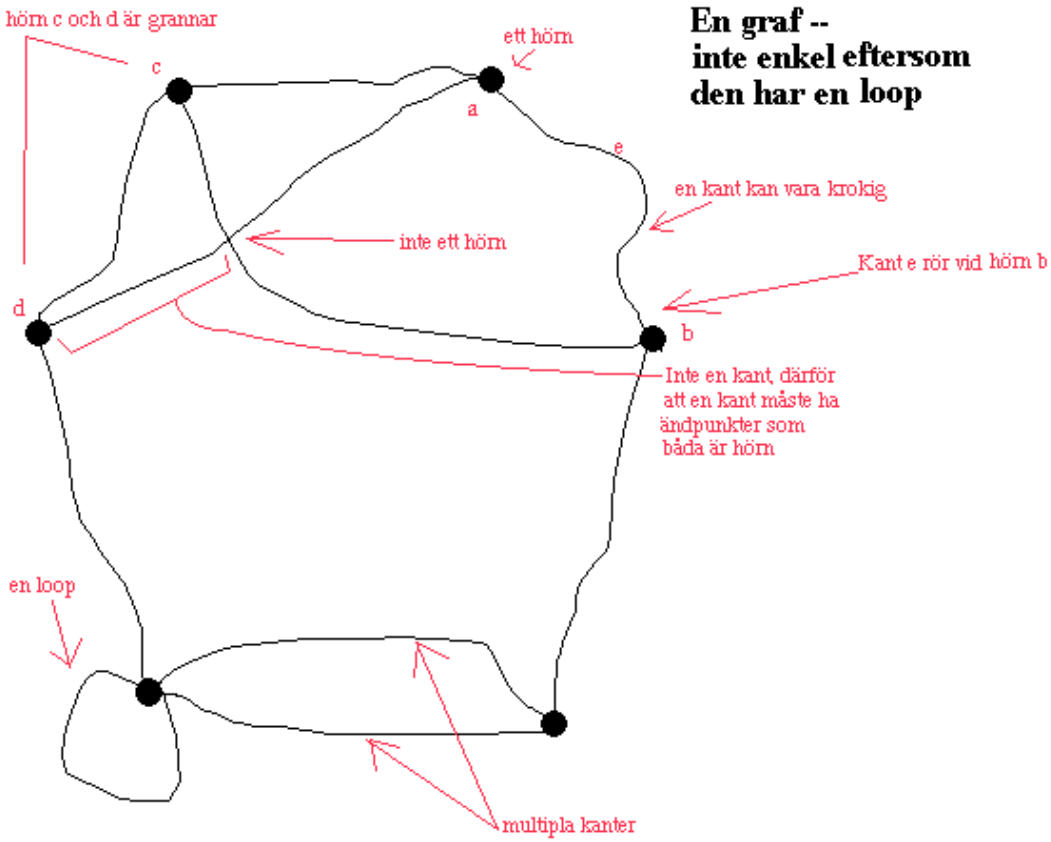
Notera också: i andra böcker kan en kant vara beskriven ab istället för (a,b) som då används för en **riktad kant** (directed edge *eller* arc) från a till b.

Introduktion och terminologi

Läs [J] Avsnitt 6.1 (6.1) [8.1] .

Mycket ny terminologi i ämnet introduceras här. En del ord som man får se upp med på sidorna 263 - 266 (304-309) [318-321] är följande: **graf** (graph), **enkel graf** (simple graph), **hörn** (vertex), **kant** (edge), **multipla kanter** (parallel/multiple edges), **loop** (loop), **grannar** (adjacent vertices/neighbours), **kanten e innehåller/rör vid hörnet v** (edge e is incident on/to vertex v), **riktad graf *eller* digraf** (directed graph *eller* digraph), **riktad kant *eller* båge** (directed edge *eller* arc), **viktad graf** (weighted graph).

Rita gärna själv en graf, en riktad graf och så vidare, och markera de olika delarna så att du får en känsla för vad de olika orden betyder. Följande skiss är en illustration av terminologin kring en graf.



Det kan vara fördelaktigt att komplettera skissen med namn på både engelska och svenska för att kunna referera till

den då du läser kapitlet.

På sidorna 270 - 271 (312-313) [325-326] introduceras några speciella klasser av grafer: **fullständiga *eller* kompletta grafer** (complete graphs) och **bipartita grafer** (bipartite graphs). Dessa är viktiga och du kommer att träffa på dem igen, så du kan kanske göra några kort med exempel på de olika typerna av grafer. Detta för att lätt kunna leta reda på vad som syftas på i nästa avsnitt utan att behöva leta efter definitionerna i boken.

Uppgift B6.1
Övningar till [J] 6.1: Uppgift 5 - 10 (1 - 6) [5 - 10].

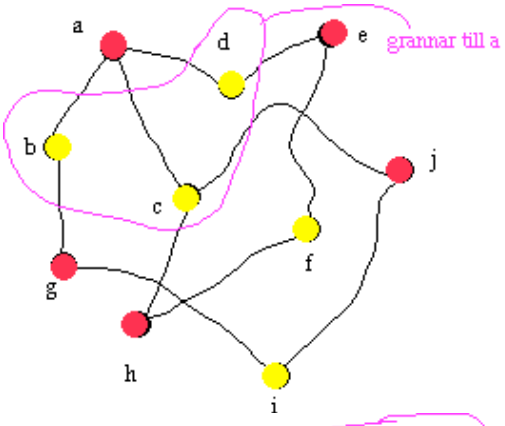
Det finns en lätt metod för att kontrollera om det är möjligt att bestämma en cykel (cycle) som börjar och slutar vid a och går genom varje kant precis en gång. Avsnitt 6.2 (6.2) [8.2] i [J] behandlar detta problem. Vill du arbeta dig fram till resultatet själv så följer lite hjälp här.

Ledning 1 Ledning 2

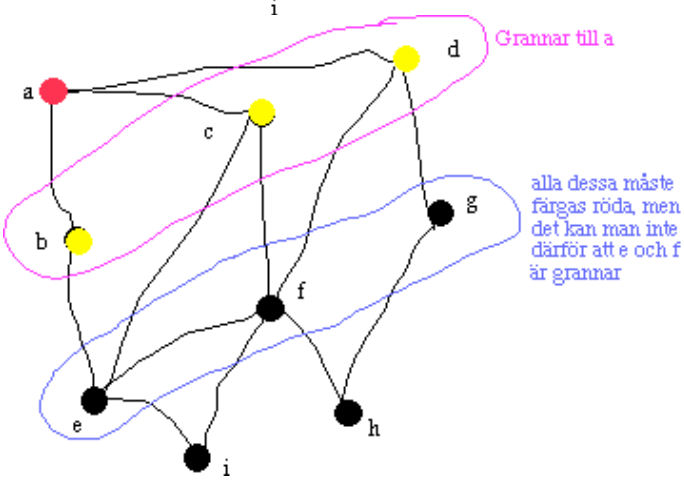
Uppgift B6.2
Övningar till [J] 6.1: Uppgift 11-16 (7-12) [11-16].

Uppgift B6.3
Övningar till [J] 6.1: Uppgift 17-23 (13-19) [17-23].
Ledning: Ett lätt sätt för att inse att en graf är bipartit är att färga hörn så att inga grannar färgas i samma färg. Kan detta utföras med endast två färger så är grafen bipartit och de disjunkta hörnmängderna ges av de två färgerna:

Exempel



Jag började vid hörn a och färgade det rött. Sedan färgade jag alla grannar till a i gul färg. Grannarna till dessa gula hörn färgades sedan röda och så vidare. Jag fick till slut en graf där två grannar alltid har olika färger. De röda hörnen bildar en mängd och de gula bildar en annan (disjunkt) mängd.



Jag började på samma sätt här, men som du kan se så finns det två hörn som bör färgas röda och som är grannar. Jag hade inget val, det går inte på något sätt att färga hörnen med bara två färger så grannar alltid får olika färger. Det innebär att denna graf inte är bipartit.

Uppgift B6.4
Övningar till [J] 6.1: Uppgift 24-25 (20-21) [24-25]

2. Stigar och cykler

Läs [J] Avsnitt 6.2 (6.2) [8.2].

Från avsnittets titel så är det klart att vi måste ta reda på vad som menas med **stigar** (paths) och **cykler** (cycles). I definition 6.2.1 (6.2.1) [8.2.1] så anges definitionen för en stig. Notera att om grafen saknar parallella kanter så räcker det att räkna upp hörnen i listan.

Uppgift B6.5
Vilka av följande är stigar i bild 6.2.1 (6.2.1) [8.2.1]?

- 1,2,5,6,2,1
- 1,2,5,7

Definitionen av en **cykel** (cycle), **enkel stig** (simple path) och **enkel cykel** (simple cycle) finns givna på sidan 276 (319) [332] i definition 6.2.14 (6.2.14) [8.2.14]. För att kontrollera att du förstår dessa definitioner gör följande uppgift med en gång:

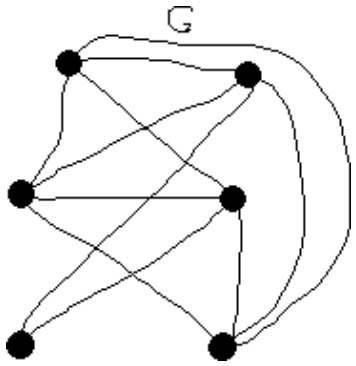
Uppgift B6.6
Övningar till [J] 6.2: 1 - 9 (1 - 9) [1 - 9].

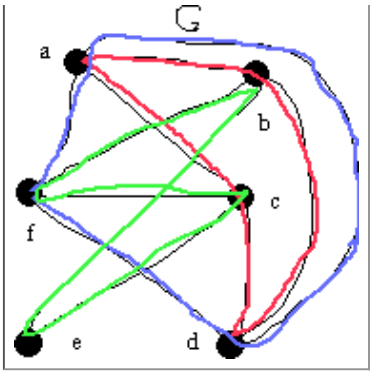
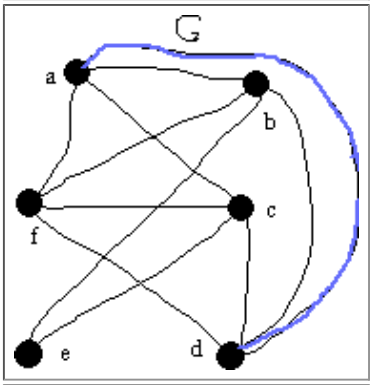
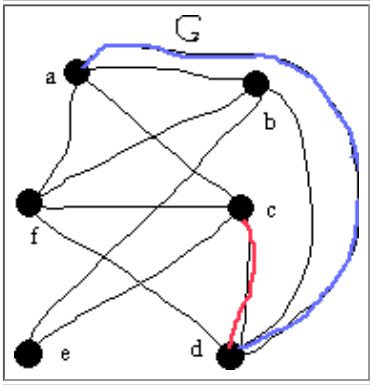
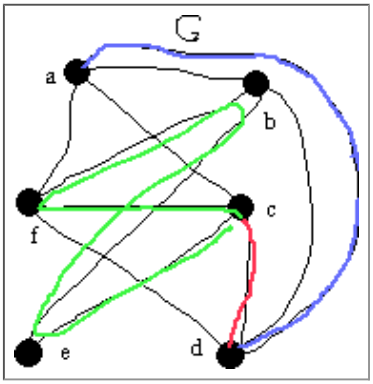
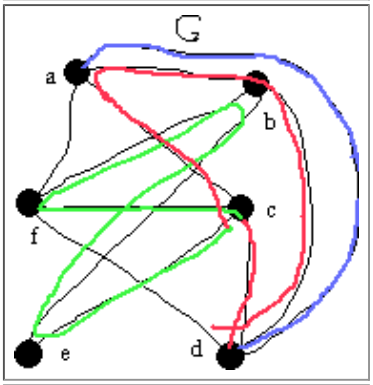
Vilka andra nya ord och begrepp diskuteras? Du bör förstå dig på **sammanhängande grafer** (connected graphs), **delgraf** (subgraph), **komponent** (component) som införs på sidorna 274 - 276 (316-319) [329-332]. Det är möjligt att en graf består av fler än en del, till exempel i en graf som representerar en ekvivalensrelation så representerar varje del en ekvivalensklass. Dessa delar kallas **komponenter** (components). En graf som består av bara én del är **sammanhängande** (connected).

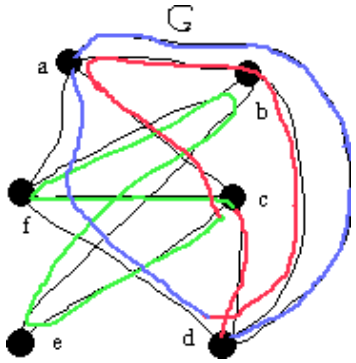
Eulercykler (Eulerian cycles)

Sidorna 277 - 280 (320-324) [332-336] behandlar problemet att bestämma en cykel i en graf som inkluderar varje kant precis en gång. Här kommer tanken bakom övningarna 5 - 10 (1 - 6) [5 - 10] i avsnitt 6.1 (6.1) [8.1] att belysas. Du kommer att få erfara att om G är en sammanhängande graf så har G en Eulercykel om och endast om varje hörn har jämn valens (Ett hörns **valens** eller **grad** (degree of a vertex) är antalet kanter med ändpunkt i hörnet (Bemärk, en loop tillför 2 till valensen). **Gradföljden** (degree sequence) för en graf är alla hörnens valenser listad i icke-avtagande ordning). Beviset av att G har en Eulercykel om och endast om varje hörn har jämn valens ges i satserna 6.2.17 (6.2.17) [8.2.17] och 6.2.18 (6.2.18) [8.2.18]. För att bevisa sats 6.2.18 (6.2.18) [8.2.18], dvs. bevisa att om alla hörn har jämn valens och grafen är sammanhängande så finns en Eulercykel, så kan vi tänka så här: Skapa en cykel utan upprepade kanter, bry dig inte om att cykeln inte tar med alla kanter. Skapa en ny cykel utan upprepade kanter med kanter som ännu inte är medtagna i den ursprungliga cykeln. Fortsätt tills att alla kanter har använts, sätt sedan ihop alla cykler för att skapa en cykel som använder alla kanter precis en gång. Detta är en process som alltid kan utföras.

Exempel

	Den ursprungliga grafen: alla hörn har jämn valens

	Här är cykel adfa (blå), cykel abdca (röd), och cykel fbecf (grön), som inte har några gemensamma kanter.
	Vi börjar vid a och följer den blå cykeln tills vi möter en cykel med annan färg. I detta fall har vi ad och sedan möter vi den röda cykeln.
	Vi börjar längs den röda cykeln tills vi stöter på en cykel med en ny färg igen, vid c.
	ännu så länge har vi adc, och nu så lägger vi till den gröna cykeln tills vi möter en ny färg. Där finns inte några nya färger, så vi lägger till hela den gröna cykeln.
	sedan resten av den röda



och slutligen resten av den blå.

Handskakningslemmat (Handshaking lemma)

Sats 6.2.21 (6.2.21) [8.2.21] kallas vanligen handskakningslemmat. Det är inte så svårt att förstå varför det är sant. När vi summerar valenserna så räknar vi bara ändpunkterna av kanterna. Varje kant har två ändpunkter, så det

totala antalet kanter = (antalet ändpunkter)/2=(summan av valenser)/2.

Detta resultat kan ibland användas till att avgöra när det är omöjligt att rita en graf, som i det följande exemplet. Antag att du vill bygga ett nätverk med fem datorer så att varje dator är kopplad till 1,1,2,3,4 av de andra datorerna respektive. Om vi ritar en graf av detta med datorer som hörn och kablar mellan datorerna som kanter, så kommer antalet datorer varje dator är kopplad till att motsvara hörnens valenser. Lägger vi ihop alla valenser, så får vi reda på antalet kanter som behövs: (1+1+2+3+4)/2 kanter. Detta är inte ett heltal så det är omöjligt att koppla ihop fem datorer på det angivna sättet. (Korollarium 6.2.22 (6.2.22) [8.2.22] kan också användas för att lösa detta problem.)

Uppgift B6.7
Övningar till [J] avsnitt 6.2 (6.2) [8.2]: 1 - 9, 10, 12, 13, 16, 22, 26, 28-33 (exhibit one = ange en), 35, 36 (samma nummer för edition 4) [samma nummer för edition 6]

Hamiltoncykler och handelsresandeproblemet

Läs [J] Avsnitt 6.3 (6.3) [8.3].

I föregående avsnitt så undersökte vi cykler som går genom varje ***kant*** precis en gång (Eulercykler). I detta avsnitt så tittar vi närmare på (enkle) cykler som går genom varje ***hörn*** precis en gång, de såkallade **Hamiltoncykler** (Hamiltonian cycles). Som vi såg så är det lätt att kontrollera om en graf har en Eulercykel. Däremot är det inte så lätt att avgöra om det finns en Hamiltoncykel. Faktiskt så finns det ***ingen*** lätt metod som alltid fungerar. Det är bara metoden i vilken man kontrollerar varje möjlig cykel som ger fullkomlig säkerhet i svaret. Att lösa problemet med denna lösningsmetod kan ta en väldigt lång tid!

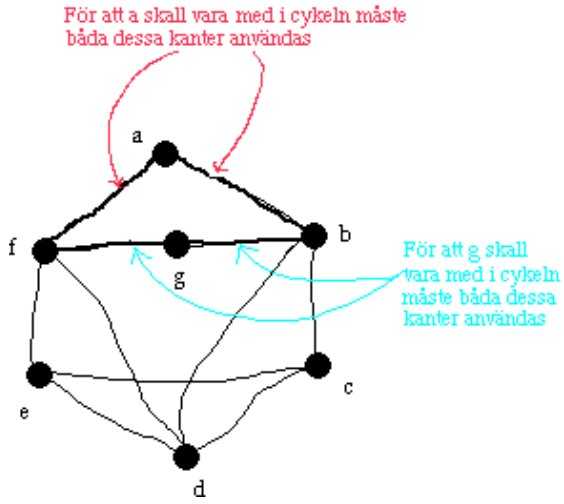
Du bör också känna till och förstå dig på **handelsresandeproblemet** (the travelling salesman problem). Läs sidorna 286 - 288 övan (332-333) [340-343].

Uppgift B6.8
Övningar till [J] avsnitt 6.3 (6.3) [8.3]: Uppg. 1-11 (1-11) [1-11].

Det är ofta svårt att avgöra om en graf har en Hamiltoncykel. Några idéer som kan hjälpa är följande.

- Är grafen bipartit? Om den är bipartit och de disjunkta hörnmängderna är av olika storlek så har grafen ingen Hamiltoncykel. Är mängderna lika stora så vet vi inte om det finns en Hamiltoncykel eller inte.
- Finns det ett hörn som delar grafen i fler än en del om vi tar bort det och alla kanter som rör vid det? Om det gör, så har grafen ingen Hamiltoncykel, ty detta hörn måste användas fler gånger än en om en cykel innehållande alla hörn ska bestämmas.
- Finns det kanter som måste användas för att nå ett hörn? Används de i cykeln kan det vara så att det är omöjligt att lägga till ytterliggare delar för att få en Hamiltoncykel, som i följande exempel.

Exempel



Dijkstras algoritm för kortaste stig

Läs [J] Avsnitt 6.4 (6.4) [8.4].

Givet två hörn i en graf så bör du kunna bestämma en kortaste stig mellan dem. Om du tycker att avsnittet är svårläst så finns det alternativa anteckningar här i html-version och här i pdf-version som kan hjälpa (anteckningarna är på engelska, sorry).

Uppgift B6.9
Övningar till [J] avsnitt 6.4 (6.4) [8.4]: Uppg. 1-5 (1-5) [1-5].

3. Grannmatrisen och incidensmatrisen för en graf

Läs [J] Avsnitt 6.5 (6.5) [8.5].

Du bör kunna det följande från avsnittet: Vi har redan sett att en graf kan representeras med en ritning eller med en lista med hörn och en lista med kanter. Här kommer ett par andra metoder. Den första är grannmatrisen för en graf och den andra är incidensmatrisen.

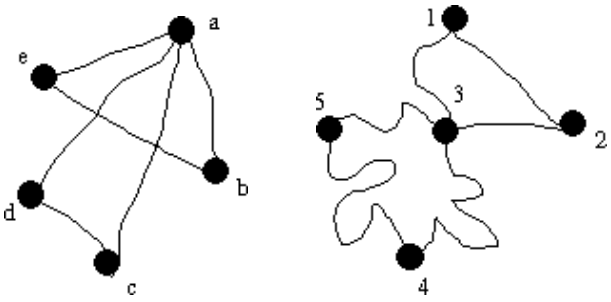
Uppgift B6.10
Övningar till [J] avsnitt 6.5 (6.5) [8.5]: Uppg. 1, 2, 3, 7, 8, 9, 13, 16, 24
(samma nummer för edition 4)[samma nummer för edition 6]

4. Grafisomorfier

Läs [J] Avsnitt 6.6 (6.6) [8.6].

Idéen bakom begreppet grafisomorfi är inte alltför svår men att faktiskt avgöra om två grafer är isomorfa är inte alltid en så enkel uppgift. Vad menar vi med isomorfi av grafer? Vi menar att två grafer representerar samma situation även om de tycks ha olika utseende.

Exempel
Följande två grafer kan båda representera 5 personer, där en person, i den första grafen a och i den andra grafen 3, känner 4 andra personer, och att e känner både b och a motsvarar i den andra grafen 1 som känner 3 och 2.



Vi kan para ihop varje hörn i den första grafen med ett hörn i den andra grafen, så att kanterna i den första grafen motsvaras av kanterna i den andra grafen. Ett sätt på vilket detta kan göras är med bijektiva funktionen

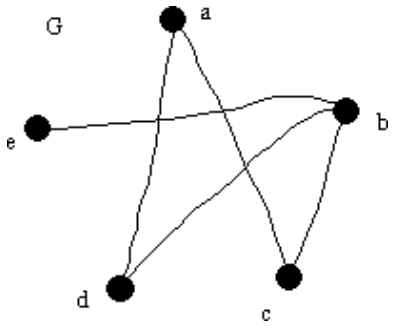
$$a \longrightarrow 3,\ b \longrightarrow 2,\ c \longrightarrow 4,\ d \longrightarrow 5,\ e \longrightarrow 1.$$

En snabb kontroll visar att alla kanterna i den första grafen motsvaras av kanter i den andra grafen. Om du kan para ihop hörn på detta sätt, så att även kanterna tas med, så är graferna **isomorfa** (isomorphic). Om det är omöjligt att göra detta så är graferna inte isomorfa.

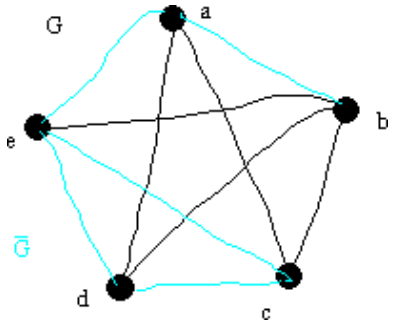
För att visa att två grafer är isomorfa så *måste* du lyckas para ihop hörnen. För att visa att två grafer inte är isomorfa så kan du visa på någon skillnad mellan graferna, som t.ex.:

- olika antal hörn
- olika antal kanter
- olika antal cykler av längd 3
- olika hörnvalens
- Om båda graferna är enkla kan det ibland vara lättare att undersöka vilka kanter som *inte* finns med, ty två grafer är isomorfa om och endast om deras komplementer är isomorfa. (**Komplementet** till en graf G innehåller alla kanter som inte finns med i G tillsammans med alla hörn i G.) Denna metod är speciellt användbar när graferna innehåller många kanter, vilket innebär att komplementgraferna har få kanter.

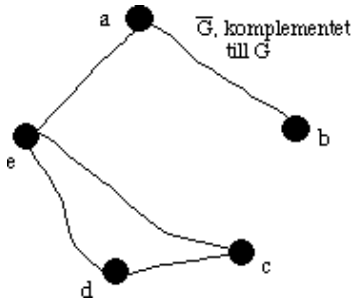
  Exempel
  Här är ett exempel på en graf och dess **komplement**.



Grafen



Grafen och dess komplement



Komplementet

Uppgift B6.11
Övningar till [J] avsnitt 6.6 (6.6) [8.6]: Uppg. 1-10 (1-10) [1-10].

5. Träd

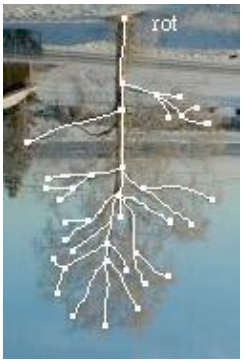
Som vi sa tidigare har vi alla stött på träd förut i form av familjeträd, utslagsturneringar och filsystemet hos datorer. Träd används också när man optimerar sökmetoder och när man analyserar antalet beslut man behöver göra i en valsituation. Vi hinner tyvärr inte titta på den här typen av problem i den här kursen.

Introduktion

Träd kan användas när man analyserar RNA, för att ta reda på släktskapet mellan olika växter, skalbaggar, virus och så vidare. Detta är ett område inom matematisk biologi. Om du är intresserad av att läsa mer om detta kan du titta på nätet på sidan The Tree of Life .

Vi har nu nämt ett antal användningsområden för träd, men vad är ett träd för någonting? Ett **träd** (tree) är en graf som är *sammanhängande* och *saknar cykler*. En graf som består av komponenter som alla är träd kallas för, ja vadå, tänk ut ett bra namn för en samling träd.

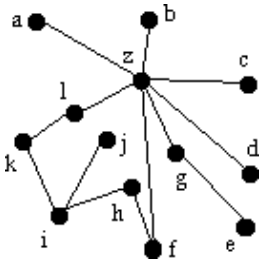
Du har troligtvis gissat rätt, men se svaret här!



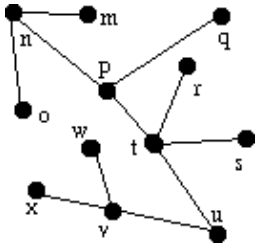
Vissa träd har ett speciellt hörn som valts som **rot** (root). Dessa träd kallas för **rotade träd** (rooted trees), och av någon anledning brukar man rita dem med roten uppåt som i bilden till vänster. Om vi fortsätter med naturkunskapen, så har rotade träd **grenar** (branches) och **löv** (leaves). Dessa begrepp introduceras i avsnitt 7.1 (7.1) [9.1].

Ett annat sätt att använda träd är i problem av följande typ. Tänk igenom vilka krav en bra lösning skall uppfylla först innan du läser avsnitt 7.3 (7.3) [9.3] och 7.4 (7.4) [9.4].

Bredbandsproblemet



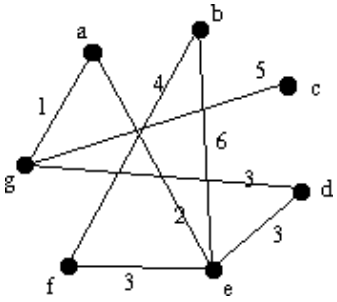
Antag att det har beslutats att alla i Sverige skall ha bredband, och att man lägger ner minsta antalet kablar som behövs. Diagrammet till vänster visar den föreslagna planen för att lägga kablarna. Är det en bra plan? Går det att reducera antalet kablar? Om det går, hur? Om inte, varför?



För en anna del av landet, har förslaget till vänster valts. Går det att reducera antalet kablar i det här fallet? Om det går, hur? Om inte, varför?

Betrakta båda exemplen, vilken typ av graf löser problemet? Tänk igenom detta en stund, och kontrollera sedan svaret i avsnittet om uppspännande träd nedan.

Viktade bredbandsproblemet



Antag att vi också vet kostnaden för varje kabel, såsom visas i grafen till höger. Talet som finns vid varje kant representerar kostnaden för den kabeln. Hur skalll man ta reda på vilka kablar man skall lägga för att minimera kostnaden? Fundera ut ett sätt som du tycker fungerar för grafen till vänster. I avsnittet om minsta uppspännande träd nedan finns det algoritmer för att lösa den här typen av problem.

Definitioner och terminologi för träd

Läs [J] Avsnitt 7.1 (7.1) [9.1].

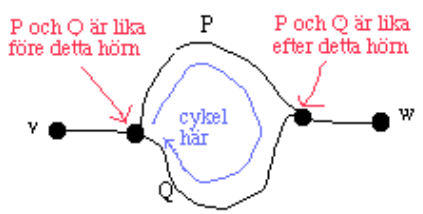
Återigen måste vi vara försiktiga med definitionerna. Vad som menas med ett 'träd' är standard, men man kan ge definitionen på flera sätt. Definition 7.1.1 (7.1.1) [9.1.1] given i avsnitt 7.1 (7.1) [9.1] ser ut så här.

- Ett **träd** T är en enkel graf som uppfyller att, om v och w är hörn i T, så finns en unik enkel stig från v till w.

Som du märkte i introduktionen, så gav vi en annan definition.

- Ett **träd** är en sammanhängande graf utan cykler.

Är den här definitionen ekvivalent med definition 7.1.1 (7.1.1) [9.1.1]? Att det inte finns några cykler innebär att det inte finns några loopar eller parallella kanter (Varför?). Definitionen av sammanhängande (definition 6.2.4 (6.2.4) [8.2.4]) säger att det finns åtminstone en enkel stig från v till w. Vi skall visa att om en graf inte innehåller några cykler, så kan det inte finnas mer än en stig mellan varje par av hörn. Vi gör detta genom att se vad som händer om det finns två enkla stigar P och Q mellan ett par av hörn som vi kallar v och w. Vi ser i diagrammet till vänster att det ger en cykel. Detta visar att det är omöjligt att inte få cykler i en graf som har två enkla stigar mellan ett par av hörn. Vi har kommit fram till att om det inte finns några cykler i en graf så medför det att det finns högst en enkel stig mellan varje par av hörn. Så båda definitionerna betyder samma sak. Till yttermera, det finns åtminstone två till sätt att definiera ett träd! Dessa finner du i sats 7.2.3 (7.2.3) [9.2.3] i avsnitt 7.2 (7.2) [9.2].



Två andra definitioner som dyker upp i avsnitt 7.1 (7.1) [9.1] är definitionerna av **höjd** (height) och **nivå** (level) i ett rotat träd. Höjden av ett träd är den maximala längden av en enkel stig som startar i roten. Nivån hos ett hörn är avståndet (antalet kanter) mellan hörnet och roten.

Då du har läst avsnitt 7.1 (7.1) [9.1] i [J] skall du veta vad följande saker betyder: träd, höjd, ett hörns nivå, rotat träd.

Uppgift B6.12
Övningar till [J] avsnitt 7.1 (7.1) [9.1] sida 330 (383) [385] i [J]:
8, 9, 10 (1, 2, 4) [8, 9, 10]

Läs [J] Avsnitt 7.2 (7.2) [9.2]. Nu mixar vi in lite familjetermer i terminologin: föräldrar, barn, förfäder och så dom gröna sakerna som växer ute. Se definition 7.2.1 (7.2.1) [9.2.1].

Som vi sa ovan så finns det olika sätt att definiera ett träd. Du skall kunna alla dessa, eftersom de belyser användbara egenskaper hos träd.

När du har läst avsnitt 7.2 (7.2) [9.2] i [J], skall du kunna

- vad följande termer betyder: förälder, barn, syskon (sibling), löv, inre hörn,
- de olika definitionerna av träd givna i sats 7.2.3 (7.2.3) [9.2.3],
- att ett träd har (antalet hörn - 1) kanter.

Uppgift B6.13
Övningar till [J] avsnitt 7.2 (7.2) [9.2]:
1, 2, 3, 4, 5, 6, 17, 19, 21, 22, 24*, 25, 26 (samma för edition 4) [samma för edition 6]

Uppspännande träd

Läs [J] Avsnitt 7.3 (7.3) [9.3].

⛔ **STOPP! Tänk igenom de två förslagen i bredbandsproblemet ovan innan du läser vidare.**

I det första förslaget ser vi att det finns en cykel

z f h i k l z.

Vi kan ta bort vilken kant som hellst från cykeln utan att dela grafen i bitar. Detta betyder att det finns åtminstone en kant för mycket i det första förslaget. I allmänhet vill vi ju inte ha några cykler i vårt förslag och vi vill att grafen skall vara sammanhängande. Det är samma sak som att säga att vi letar efter ett träd som innehåller alla hörn i grafen. Ett sådant träd kallas för ett **uppspännande träd** (spanning tree).

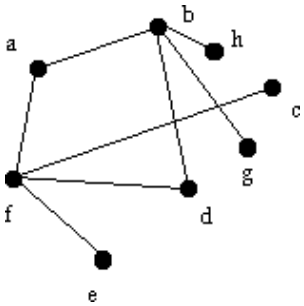
I det andra förslaget har vi redan ett träd. Om vi tar bort en kant kommer inte grafen längre att vara sammanhängande, vilket betyder att ett träd är det bästa resultatet. Hur många kablar behövs? Exakt antalet kanter i trädet, som är samma antal som antalet hörn minus 1. (Observera: Om vi tar bort en kant från ett träd, så blir resultatet en skog bestående av två träd.)

Finns det alltid en lösning till den här typen av problem? Ja, så länge som grafen man utgår ifrån är sammanhängande. Hur hittar man ett uppspännande träd?

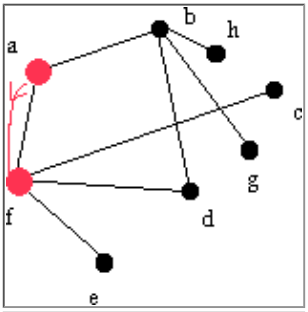
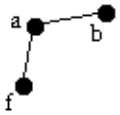
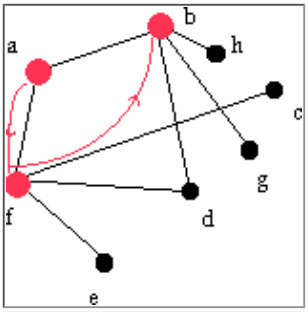
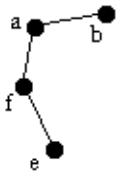
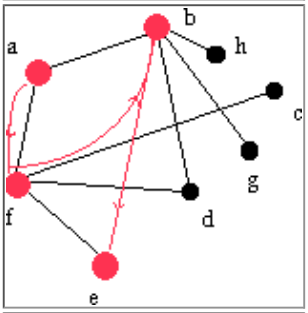
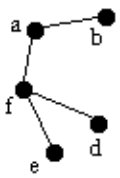
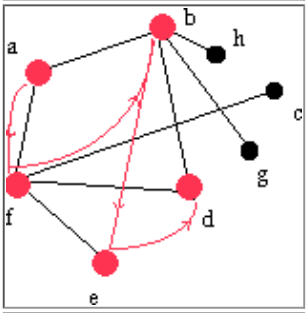
Bredd före djup- och Djup före bredd-sökning

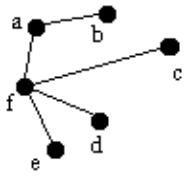
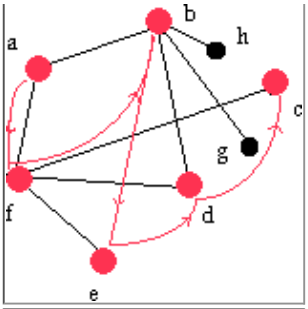
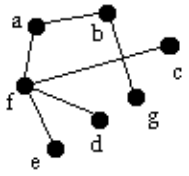
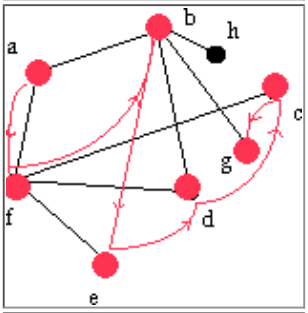
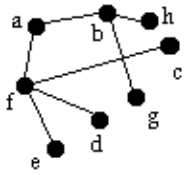
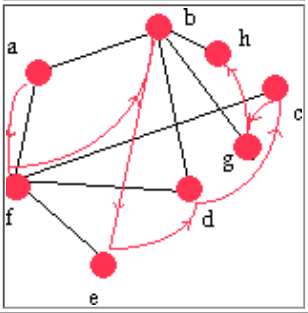
Vi skall nu titta på två metoder att hitta ett uppspännande träd. Båda startar vid ett valt hörn. Följande diagram och tabeller ger ett exempel på hur dessa metoder fungerar.

Betrakta grafen:

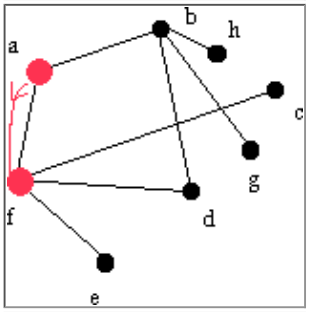


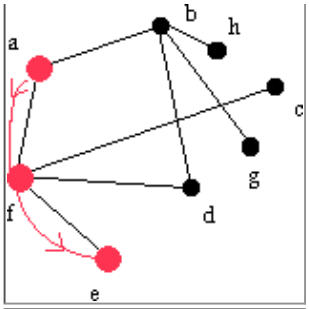
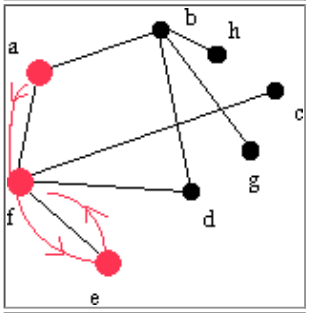
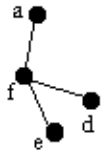
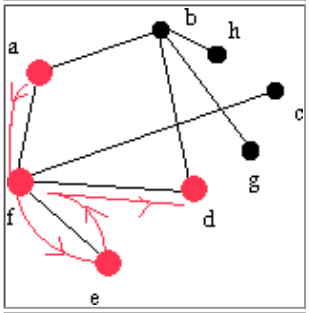
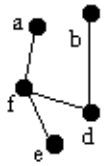
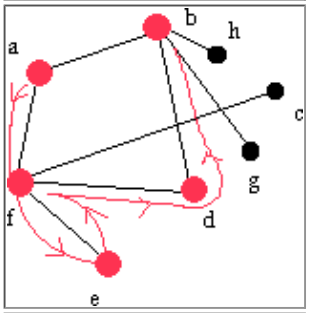
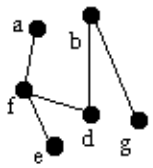
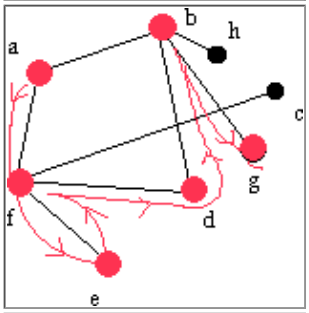
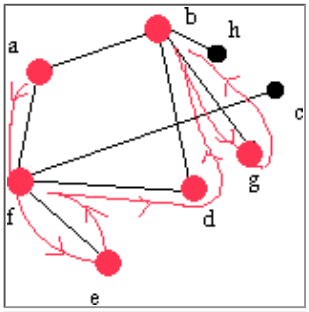
I båda metoder startar vi sökningen vid hörn a.

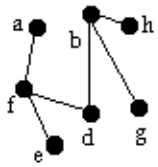
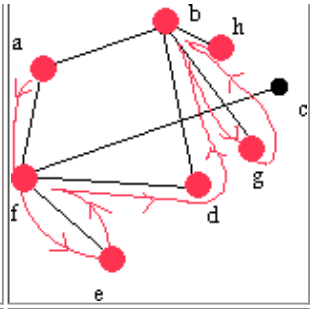
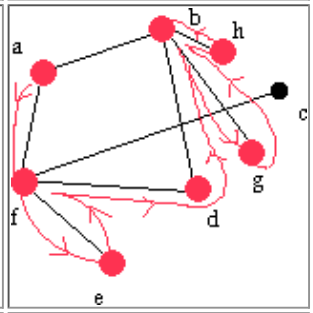
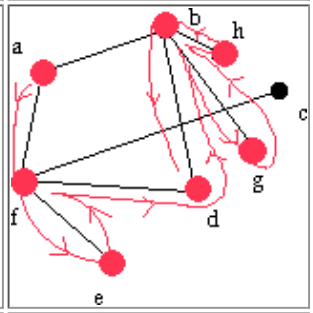
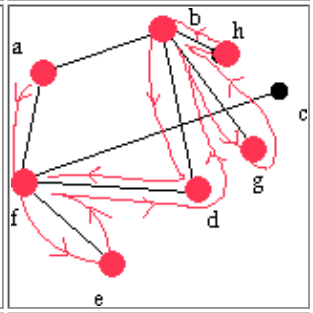
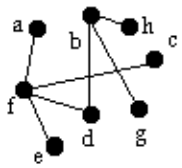
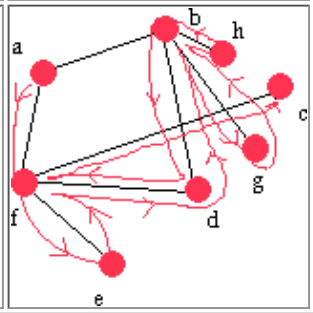
Bredd före djup-sökning							
Lista över hörnen	Söker vid	Lägg till hörn till lista	Ta bort hörn från lista	Lägg till kant till träd	Trädet så långt	Sökväg så långt	Kommentarer
a	a	f	-	af			Lägg till f till slutet på listan. Leta alltid efter grannar till det ***första*** hörnet i listan. I det här fallet, leta efter grannar till a och fortsätt tills det inte finns några kvar.
af	a	b	-	ab			Fortsätter att söka efter grannar till a.
afb	a	-	a	-	-	-	Det finns inga fler grannar till a så vi tar bort a från listan, och fortsätter att söka vid nästa hörn i listan.
fb	f	e	-	fe			Söker efter grannar till f.
fbe	f	d	-	fd			Söker efter grannar till f.

fbed	f	c	-	fc			Söker efter grannar till f.
fbedc	f	-	f	-	-	-	Det finns inga fler grannar till f, så vi tar bort det från listan.
bedc	b	g	-	bg			Söker efter grannar till b.
bedcg	b	h	-	bh			Nu har vi undersökt alla hörn i grafen och vi kan avsluta sökningen.

I en del fall kan det hända att listan med hörn blir tom innan man undersökt alla hörn. Detta betyder att grafen måste vara icke-sammanhängande, den består alltså av mer än en komponent. I sådana fall hittar man ett uppspännande träd för varje komponent, dessa utgör då en uppspännande skog för grafen.

Djup före bredd-sökning							
Lista över hörnen	Söker vid	Lägg till hörn till lista	Ta bort hörn från lista	Lägg till kant till träd	Trädet så långt	Sökväg så långt	Kommentarer
a	a	f	-	af			Återigen lägger vi till f till listan. Med den här metoden söker vi vid det ***sista*** hörnet i listan så vi fortsätter att söka vid f.

af	f	e	-	fe			Vi söker efter en granne till f, som vi lägger till i listan.
afe	e	-	e	-	-		Vi söker efter en granne till e, men det finns ingen, så ta bort e från listan och gå tillbaka till f för nästa steg.
af	f	d	-	fd			Vi söker efter en ny granne till f och hittar d.
afd	d	b	-	db			Vi söker vid d och hittar b.
afdb	b	g	-	bg			Vi söker vid b och hittar g.
afdbg	g	-	g	-	-		Vi söker vid g och hittar inga nya grannar, så vi går tillbaka till b igen för nästa steg.

afdb	b	h	-	bh			Vi söker vid b och hittar h.
afdbh	h	-	h	-	-		Det finns inga nya grannar vid h, så vi går tillbaka till b igen.
afdb	b	-	b	-	-		Det finns inga nya grannar till b, så vi går tillbaka till d.
afd	d	-	d	-	-		Det finns inga nya grannar till d, så vi går tillbaka till f.
af	f	c	-	fc			Antingen kan vi stanna här eftersom alla hörn finns med i trädet, eller så fortsätter man tills listan är tom.

När du har läst avsnitt 7.3 (7.3) [9.3] i [J], skall du kunna

- vad följande termer betyder: uppspännande träd (spanning tree)
- hur man genomför en bredd före djup-sökning
- hur man genomför en djup före bredd-sökning

Uppgift B6.14
Övningar till [J] avsnitt 7.3 (7.3) [9.3]:
1, 4, 7, 9 (1, 4, 7, 9) [1, 2, 7, 9]

6. Minsta uppspännande träd

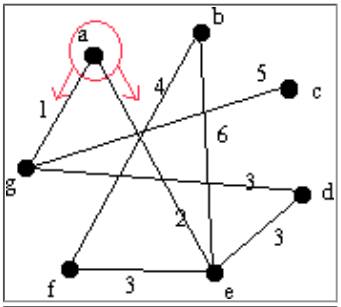
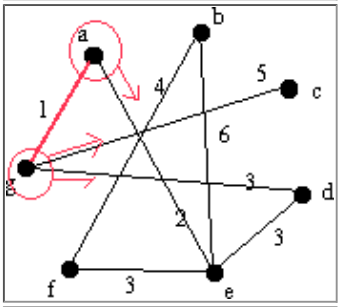
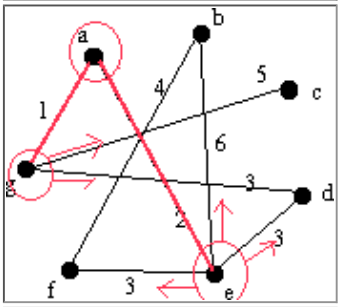
Läs [J] Avsnitt 7.4 (7.4) [9.4].

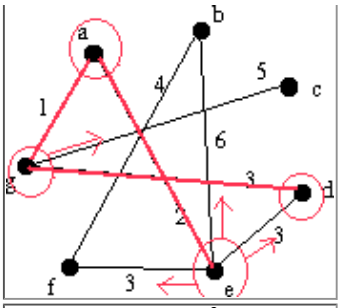
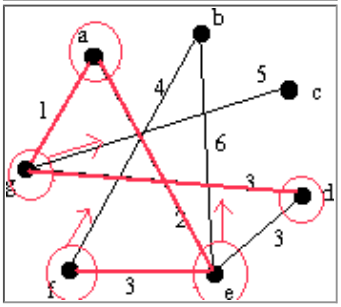
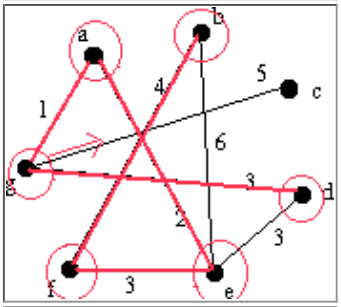
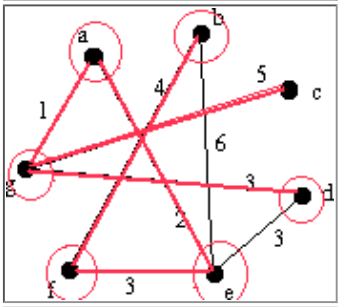
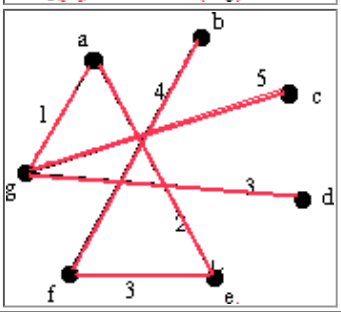
STOPP! Tänk igenom den viktade versionen av bredbandsproblemet ovan innan du läser vidare.

Precis som förrut vill vi hitta ett uppspännande träd. I det här fallet vill vi dessutom ha det till minsta möjliga kostnad. Ett sådant träd kallas för ett **minsta uppspännande träd**. Det finns två metoder i avsnitt 7.4 (7.4) [9.4] för att lösa detta problem, **Prims algoritm** är en metod och den andra är **Kruskals algoritm**. En fråga man kan ställa sig är 'Vilken är bättre?' Svaret varierar beroende på antalet kanter och antalet hörn. Du behöver inte lära dig båda metoderna i den här kursen, välj den du tycker är bäst om du inte vill lära dig båda. Här finns ett exempel av varje algoritm för bredbandsproblemet.

Prims Algoritm

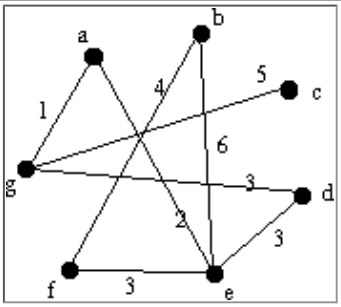
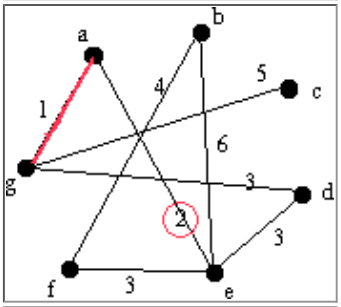
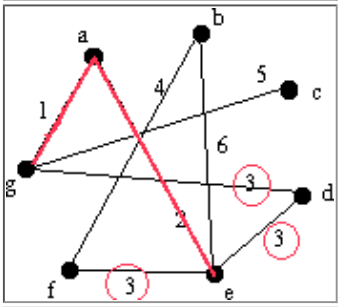
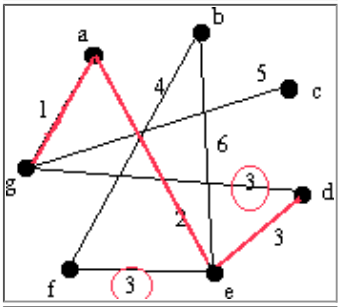
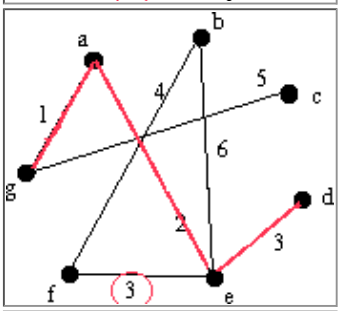
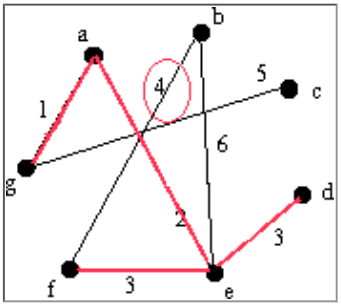
Exempel
Se algoritm 7.4.3 (7.4.3) [9.4.3]:

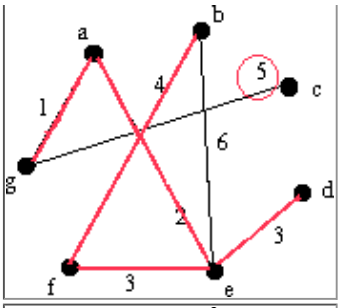
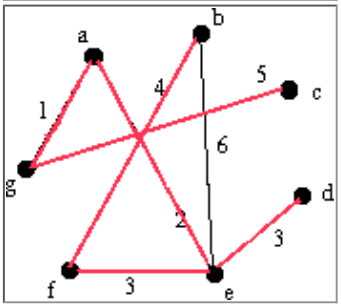
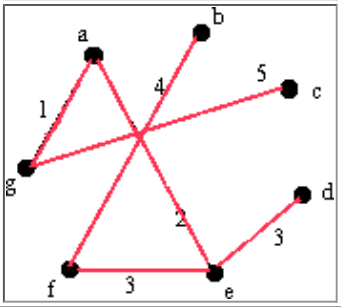
	Välj ett hörn att starta vid. Vi väljer a. Vi drar en cirkel kring detta hörn och lägger in det i en mängd S. Kontrollera alla kanter som rör vid a och välj kanten med minsta vikten. Den här kanten kommer att vara en del av det minsta uppspännande trädet. I det här fallet väljer vi kanten *ag* av vikt 1. Vi lägger till hörnet g till S så S={a, g} och drar en cirkel omkring g.
	Sen undersöker vi alla kanter som har exakt en ändpunkt i S, i det här fallet kanter med ändpunkter i a eller g. (Markerade med pilar.) Vi väljer kanten med lägst vikt, alltså *ae* av vikt 2. Vi lägger sedan till e till S så S={a, g, e}
	Återigen kontrollerar vi alla kanter som har en ändpunkt i S, och väljer den med lägst vikt. Den här gången kan vi välja mellan *gd*, *ef* och *de*. I en dator skulle man ha kanterna ordnade och man skulle ha valt den första i listan. Här väljer vi bara en, t.ex. *gd*. S={a, g, d, e}

	Nu behöver vi inte längre kolla ed eftersom både e och d är i S. Vi väljer *ef* av vikt 3. S={a, g, d, e, f}
	Sen väljer vi *fb* av vikt 4. S={a, g, d, e, f, b}
	Det finns bara en kant kvar mellan hörn i S och hörn som inte är i S, och det är *gc*.
	Alla hörn finns nu i S, så vi har det minsta uppspännande trädet markerat i grafen.
	Ta bort alla extra kanter och cirklarna så har vi det minsta uppspännande trädet. Detta träd har vikt 1+2+3+3+4+5=18.

Kruskals Algoritm

Exempel
Läs texten före övning 20 (20) [20]:

	Vi söker reda på kanten med minsta vikten och lägger till den till en mängd E. Kant ag har lägst vikt så E={ag}. Vi färgar kanten ag röd.
	Nu letar vi reda på den kant med lägst vikt som finns i den svarta delen av grafen. Det finns en kant av vikt 2 och det är ae. Vi kollar så att ae inte bildar en cykel med någon av kanterna i E. Det gör den inte så vi lägger till ae till E, så E={ag, ae}
	Nu letar vi reda på nästa kant med lägst vikt. Det finns tre kanter av vikt 3, gd, ed och ef. Vi väljer en, t.ex. gd, och kollar så att det inte bildas några cykler med kanter i E. Det gör det inte, så vi lägger till kanten till E. Nu är E={ag, ae, gd}.
	Nu kollar vi en annan kant av längd 3 nämligen ed. Denna bildar en cykel med kanterna i E nämligen aedga. Eftersom vi vill ha ett träd, lägger vi inte till den här kanten, utan tar bort den från grafen.
	Den sista kanten av vikt 3 är ef, och den ger ingen cykel så vi lägger till den till E, så att vi får E={ag, ae, gd, ef}.
	Vi lägger till bf till E.

	Vi lägger till gc till E.
	Nu har vi 6 kanter i E och G har 7 hörn. Eftersom vi vet att ett träd har (antal hörn - 1) kanter, har vi rätt antal kanter för ett träd. Vi har hittat ett minsta uppspännande träd. Det har sina kanter i E och sina hörn i G.
	Ett minsta uppspännande träd av vikt 18.

När du har läst avsnitt 7.4 (7.4) [9.4] i [J], skall du

- veta vad följande termer betyder: minsta uppspännande träd (minimum spanning tree),
- kunna utföra *antingen* Prims *eller* Kruskals algoritm,
- kunna beskriva *antingen* Prims *eller* Kruskals algoritm.

Uppgift B6.15
Övningar till [J] avsnitt 7.4 (7.4) [9.4]:
1,2,3,4,5 (1,2,3,4,5) [1,2,3,4,5] (Välj antingen Prims eller Kruskals algoritm.)

Övningsuppgifter

Uppgift B6.1 - B6.15 i texten ovan.

The author welcomes comments and corrections via email. All contributions incorporated in updates of the manuscript will be acknowledged.

The author would like to thank the following for their contribution to various updates of the original manuscript:
Sam Lodin, Katharina Huber and Pia Heidtmann.

Current lecturers:
Pia Heidtmann, Sam Lodin.

c/o Pia Heidtmann
MID SWEDEN UNIVERSITY
Department of Engineering, Physics and Mathematics
Mid Sweden University
S-851 70 SUNDSVALL
Sweden

Updated 070811