UNDERVISNINGS
MINISTERIET
STYRELSEN FOR
UNDERVISNING OG KVALITET

Matematik A

Studentereksamen

Ny ordning

Forberedelsesmateriale

ny-stx191-MAT/A-24042019

Onsdag den 24. april 2019

Forberedelsesmateriale til stx-A MATEMATIK

Der skal afsættes 6 timer af holdets sædvanlige uddannelsestid til, at eleverne kan arbejde med forberedelsesmaterialet forud for den skriftlige prøve.

Ved den skriftlige prøve kan indhold og metoder fra forberedelsesmaterialet indgå i opgaver i begge delprøver.

Oplægget indeholder teori, eksempler og øvelser i tilknytning til et emne, der ligger umiddelbart i forlængelse af emnerne i kernestoffet. I dette forberedelsesmateriale er emnet ”Grafteori”.

Alle hjælpemidler er tilladt, og det er tilladt at modtage vejledning under arbejdet med dette forberedelsesmateriale.

Resultaterne af arbejdet med dette forberedelsesmateriale bør medbringes til delprøve 2 af den skriftlige prøve.

Det foreliggende materiale er gældende for eksamen maj-juni 2019, august 2019 og december 2019.

Indhold

Indledning

I forberedelsesmaterialet er der både øvelser og opgaver. Øvelserne er tænkt som hjælp til forståelse af teorien, herunder beviser for nogle af sætningerne. Opgaverne er tænkt som forberedelse til de opgaver, der kommer til den skriftlige eksamen.
I forberedelsesmaterialet anvendes fem typer af farvede bokse. Se eksempler på farvekoderne her:

Definition

Eksempel

Sætning

Øvelse	Øvelserne er tænkt som hjælp til forståelse af teorien, herunder også beviser for nogle af sætningerne.

Opgave	Opgaverne er tænkt som forberedelse på de opgaver, der kan komme til den skriftlige eksamen.

Opgave 	Opgaverne markeret med hånd og blyant kan kun forekomme i delprøve 1.

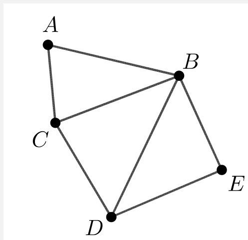
Opgave 	Opgaverne markeret med et tastatur kan kun forekomme i delprøve 2.

Grafteori

Grafteori er ikke et nyt matematisk emne, men med vores store brug af computere er emnets aktualitet forøget. De algoritmer, man finder i grafteori, gennemføres ofte med computeren, og grafteori er således et vigtigt emne inden for den diskrete matematik. Dette forberedelsesmateriale lægger dog ikke op til computerbrug men snarere til en fordybelse i selve algoritmerne. Begrebet *graf* har i denne sammenhæng en ganske anden betydning, end den vi kender fra arbejdet med funktioner.

Grafteori kan bl.a. benyttes til at give en forenklet fremstilling af problemer, som fx vedrører optimering af forskellige netværk. Det kan være spørgsmålet om, hvordan man finder den hurtigste rute mellem to destinationer (tænk f.eks. på Google Maps), den billigste flyrute mellem to byer, eller hvordan man planlægger etablering af et fibernet billigst. Vi kommer i dette materiale til at arbejde med to forskellige algoritmer: Prims algoritme og Dijkstras algoritme. Desuden introduceres de grundlæggende grafteoretiske begreber samt begreberne Eulertur og Eulergraf, og vi ser på deres anvendelser.

Eksempel 1 Figuren viser en skitse af vejsystemet i en park, hvor en motionist løber sin daglige motionstur. Bogstaverne A, B, C, D og E angiver vejkryds.



Løberen ønsker at løbe på alle veje, men kun én gang på hver vej. Dette kan lade sig gøre, hvis han fx løber ad ruten
$C \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow B \rightarrow E \rightarrow D$.

Bemærk, at løberen ikke slutter samme sted, som han startede.

Øvelse 1

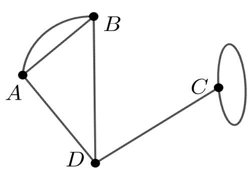
a) Lav en anden løberute på figuren i eksempel 1, hvor løberen kommer rundt på alle vejene præcis én gang? Gem resultatet, da du skal se på det senere.

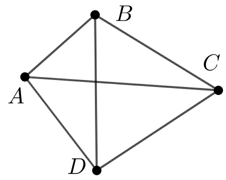
b) Løberen ønsker at starte og slutte i punkt C. Han behøver ikke at nå rundt på alle veje, men ønsker at undgå at komme forbi det samme vejkryds mere end én gang. Opskriv en mulig rute. Hvor mange forskellige ruter kan du finde?

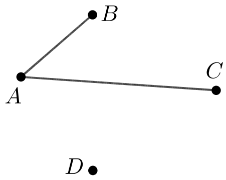
c) Er det muligt at lave en rute, hvor man starter og slutter samme sted, og som samtidig kommer rundt på alle veje præcis én gang?

Figuren i eksempel 1 er et eksempel på en *graf*.

En graf består af en samling af *hjørner*, som på figuren i eksempel 1 angives med punkterne A, B, C, D og E. *Kanterne* i en graf er stregerne, der forbinder hjørnerne parvist. Man kan godt forestille sig to hjørner, der forbindes af mere end én kant, og tilsvarende behøver to hjørner ikke nødvendigvis at være forbundet af en kant. Et hjørne kan også være forbundet med sig selv via en kant. I dette tilfælde kaldes det for en *løkke*.







Ovenstående figurer viser eksempler på grafer med hjørnerne A, B, C og D, men med forskellige kanter: Figuren til venstre viser en graf med en løkke ved hjørne C, hjørnerne A og B er forbundet af to kanter, mens der ikke findes en kant, som forbinder hjørnerne B og C, og der er heller ingen kant mellem A og C. Figuren i midten viser de samme 4 hjørner, men nu med kanter mellem alle hjørnerne. Her er der ingen *dobbeltkanter* eller løkker. Figuren til højre viser de 4 hjørner, og enkelte kanter. Grafen siges at være *ikke sammenhængende*, da der ikke er nogen kant, der forbinder hjørne D med de øvrige hjørner.

Øvelse 2

a) Tegn en graf med hjørnerne A, B, C, D og E og med kanterne AB, BC, DE og BD.

b) Tegn en graf med de samme hjørner, hvor alle hjørnerne er forbundet af kanter, og hvor to af hjørnerne er forbundet af to kanter.

c) Tegn en graf med de samme hjørner. Grafen skal have en løkke ved et hjørne, og grafen skal være ikke-sammenhængende.

Vi kan nu definere nogle centrale begreber indenfor grafteori:

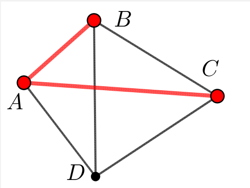
Definition 1 **Graf og delgraf**

En *graf* G består af to mængder: En mængde af hjørner og en mængde af kanter.

En *delgraf* af en graf G består af nogle af hjørnerne i G og nogle af kanterne mellem de udvalgte hjørner i G.

Eksempel 2

Figuren viser en graf med fire hjørner. En delgraf med hjørnerne A, B og C er markeret. Delgrafen indeholder kanterne AB og AC, mens kanten BC ikke er med, selvom hjørnerne B og C er.

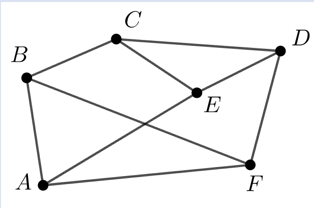


Delgrafer behøver ikke at være sammenhængende. For eksempel er delgrafen bestående af hjørnerne A, B og C og som kun har kanten AB også en delgraf.

Øvelse 3

Figuren viser en graf.

Indtegn 3 forskellige delgrafer på grafen.



Vi får brug for nogle begreber, der beskriver forskellige typer af delgrafer på en graf.

Definition 2 **Vandring, tur og sti**

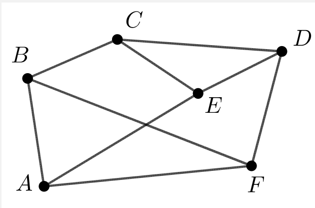
En *vandring* på en graf er en sekvens af hjørner, der er forbundet af kanter.

En *tur* er en vandring, der består af kanter, der hver kun kan være med én gang.

En *sti* er en tur, hvor samme hjørne ikke besøges flere gange.

Eksempel 3

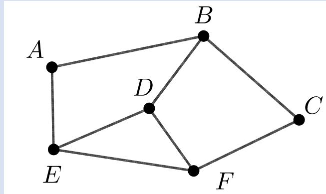
Delgrafen $B \to C \to E \to C \to D$ er et eksempel på en vandring. Det er ikke en tur, da kanten CE er med to gange.



Delgrafen $C \to E \to D \to C \to B$ er et eksempel på en tur. Det er ikke en sti, da hjørne C er med to gange.

Delgrafen $A \to E \to D \to C \to B \to F$ er et eksempel på en sti.

Øvelse 4 Figuren viser en graf.



a) Opskriv en sekvens af hjørner, der angiver en vandring, som ikke også er en tur.

b) Opskriv en tur, der ikke også er en sti.

c) Opskriv en sekvens af hjørner, der angiver en sti.

Definition 3 **Kreds**

En *kreds* er en tur, der starter og slutter i det samme hjørne, mens de andre hjørner er forskellige. Starthjørnet/sluthjørnet er således med to gange, mens de andre hjørner er med præcis én gang.

Eksempel 4 Delgrafen $A \to B \to D \to E \to A$ fra figuren i øvelse 4 er et eksempel på en kreds.

Øvelse 5 Hvor mange kredse kan du finde på figuren i øvelse 4?

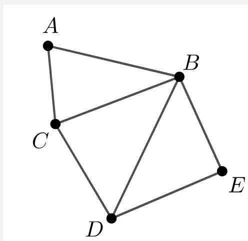
Øvelse 6 Billedet viser et rekreativt område, hvor områdets veje kan udgøre kanterne i en graf, og hvor hjørnerne er tegnet ind på billedet.

a) Skitsér kortet i en graf.

b) Hvor mange kredse, der starter og slutter i hjørne H, kan du finde på grafen?

c) Indtegn en kreds, der indeholder alle hjørner i grafen, og som starter og slutter i hjørne H.

Eksempel 5 Vi vender nu tilbage til motionisten fra eksempel 1. Motionistens overvejelser om løberute kan formuleres ved hjælp af begreber fra grafteorien. Ruten $C \to A \to B \to C \to D \to B \to E \to D$ er et eksempel på en tur, der ikke er en sti, da flere hjørner besøges flere gange.



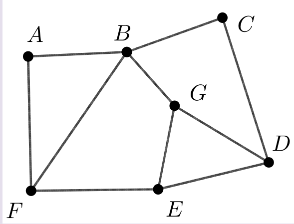
En løberute, der starter og slutter samme sted, og som ikke besøger de andre hjørner eller kanterne mere end én gang, vil være et eksempel på en kreds.

Øvelse 7 Formulér de tre delopgaver i øvelse 1 med begreberne fra grafteori.

Opgave 1 Figuren viser en graf.

a) Angiv en sti i grafen.

b) Angiv to kredse i grafen.

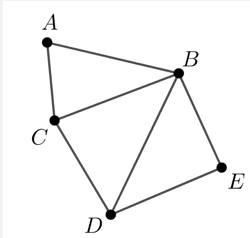


For at kunne undersøge løberens problemer med at finde ruter, hvor han ikke skal løbe på de samme veje flere gange, indfører vi nu et nyt begreb: Et hjørnes *grad*.

Definition 4 **Graden af et hjørne**

Antallet af kanter, som udgår fra et hjørne, kaldes hjørnets *grad*.

Eksempel 6 Hjørnerne A og E på figuren har graden 2, mens hjørne C og D har graden 3, og hjørne B har graden 4.



Grafen i eksempel 6 stammer fra eksemplet med løberen. Vi så i øvelse 1, at der var op til flere ruter, der havde alle kanter med netop én gang.

Øvelse 8 Find dit resultat fra øvelse 1. Kan du finde en sammenhæng mellem graden af hjørnerne og starthjørnerne for de ruter, du fandt? Se også på sluthjørnerne.

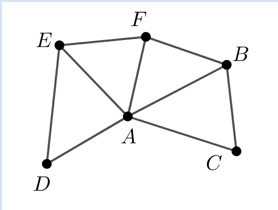
Øvelse 9

a) Angiv graden af de 6 hjørner på grafen på figuren.

b) Overbevis dig selv om, at der ikke findes en tur, der indeholder alle kanterne i grafen.

c) Kan du fjerne en kant, så du kan lave en tur, der indeholder alle (de resterende) kanter?

d) Angiv graden af hjørnerne i den nye graf.

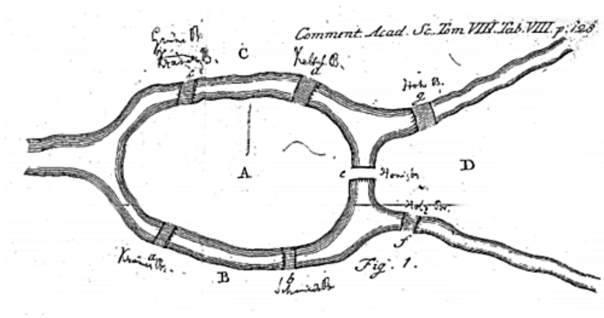


Det viser sig, at der findes en sammenhæng mellem graden af hjørnerne i en graf og muligheden for at lave en tur, der indeholder alle kanter. Dette ser vi på i næste afsnit.

Eulergrafer

Et klassisk eksempel på grafteori omhandler broerne i byen Königsberg (hedder nu Kaliningrad) og går helt tilbage til 1700-tallet.
Nedenfor ses et kort over broernes placering i forhold til byen. Kortet er tegnet i 1736 af datidens store matematiker Leonard Euler og viser landområderne A, B, C og D samt de syv broer. Euler formulerede problemet: Kan man lave en rute for en spadseretur i Königsberg, der krydser alle 7 broer netop én gang?

Eulers gode idé var at oversætte broer til streger og landområder til punkter, som i grafteoriens sprog bliver til kanter og hjørner.



Øvelse 10 Tegn en graf, som kan bruges til at beskrive problemet om broerne i Königsberg, idet du lader broerne svare til kanter og landområder svare til hjørnerne A, B, C og D. Gem dit resultat til senere brug.

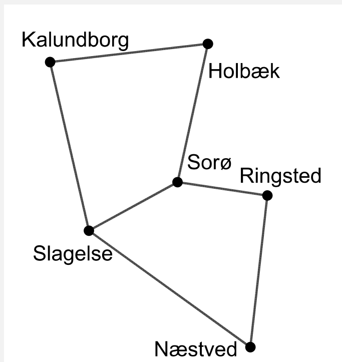
Problemet omkring spadsereturen i Königsberg svarer til eksemplet med løberen (se øvelse 1), der ønsker at løbe på alle vejene i parken netop én gang. Problemet giver anledning til følgende definition:

Definition 5 Hvis der findes en tur, som gennemløber alle kanter i grafen, kaldes turen for en *Eulertur*.

Hvis der findes en Eulertur, som starter og slutter samme sted, kaldes denne for en *lukket Eulertur*, og grafen kaldes en *Eulergraf*.

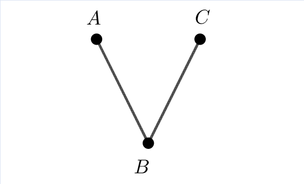
Vi skal i det følgende analysere hvilke kriterier, der skal være opfyldt, for at man kan danne en Eulertur på en graf, eller for at der er tale om en Eulergraf. Afsnittet er bygget op omkring et eksempel om et vejnet, der introduceres i eksempel 7.

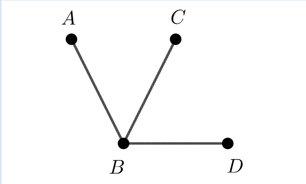
Eksempel 7 Vi ser på et udsnit af vejnettet i Danmark fra www.krak.dk, som viser hovedvejnettet mellem byerne Næstved, Ringsted, Sorø, Slagelse, Holbæk og Kalundborg.
Da vi kun er interesseret i et billede af *vejnettet* mellem de seks byer, kan vi tegne en skitse af kortet vha. en graf, hvor hjørnerne repræsenterer byerne, og kanterne illustrerer, at der er en vejforbindelse mellem to byer.

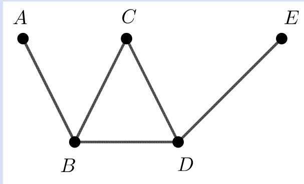


Vejvæsenet skal kontrollere vejnettet mellem de seks byer, så vi ønsker at finde en rute, hvor hver enkelt vejstrækning kontrolleres netop én gang af hensyn til økonomien. Hver kant på skitsen skal altså tilbagelægges netop én gang på køreturen svarende til, at vi skal finde en Eulertur.
Vi kan hurtigt overbevise os selv om, at det ikke er muligt, hvis køreturen starter i én af byerne Kalundborg, Holbæk, Næstved eller Ringsted. Overvej dette! Hvis vi derimod starter køreturen i Sorø eller Slagelse, kan vi komme rundt på alle veje netop én gang, uden dog at komme tilbage til startstedet. Fx er Sorø $\rightarrow$ Holbæk $\rightarrow$ Kalundborg $\rightarrow$ Slagelse $\rightarrow$ Næstved $\rightarrow$ Ringsted $\rightarrow$ Sorø $\rightarrow$ Slagelse en mulig rute. Vi kan altså finde en Eulertur på grafen.
Vi slutter ikke køreturen i startstedet, så vi har ikke fundet en lukket Eulertur.

Øvelse 11 Nedenfor er der givet tre forskellige skitser af et vejnet mellem henholdsvis tre, fire og fem byer.







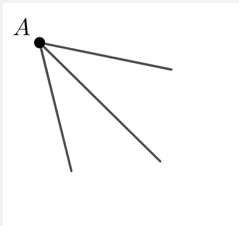
Undersøg, om der findes en Eulertur eller en lukket Eulertur på graferne.

Vi kan for hvert vejnet prøve at løse opgaven for vejvæsenet ved simpelthen at prøve os frem. Men vi kunne også lede efter nogle egenskaber ved et vejnet, som kan gøre det muligt for os let at løse vejvæsenets opgave.

Vi zoomer ind på en by på grafen og opdeler situationen i to tilfælde, alt efter om der udgår et lige eller et ulige antal veje fra byen.

Eksempel 8 Vi ser på by A, hvortil der er tre veje.

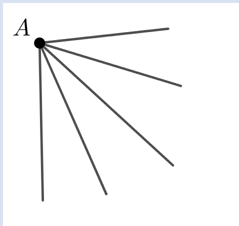
Strategi 1: (start i A)
Vi starter i A, og vi forlader byen ad en kant. Derefter skal vi komme tilbage ad en anden kant og forlade A ad den tredje kant.



Strategi 2: (start i et andet punkt end A)
Vi starter ikke i A, dvs. vi kommer til A ad en kant og forlader A ad en anden kant. Vi mangler så at køre ad den sidste kant, og dermed skal løsningen for vejvæsenet slutte i A.

Konklusion: Hvis hjørne A har grad 3 og vi starter i A, så slutter vi ikke i A. Men omvendt gælder der: Hvis vi ikke starter i A, så slutter vi i A.

Øvelse 12 Betragt en by A, hvortil der er fem veje.

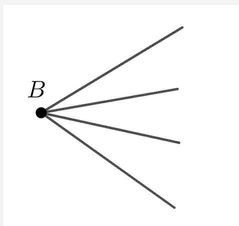


a) Argumentér for, at vi får samme konklusion, som i tilfældet med tre veje (jf. eksempel 8).

b) Generalisér til situationen, hvor byen A har et ulige antal veje.

Eksempel 9 Betragt en by B, hvortil der er fire veje.

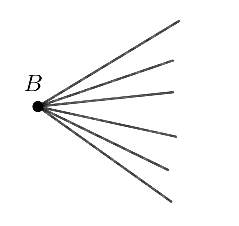
Strategi 1: (start i B)
Vi starter i B, dvs. vi forlader byen ad en kant. Derefter kommer vi tilbage ad en anden kant, og vi forlader igen B ad en tredje kant. Når vi bruger den fjerde kant kommer vi til at slutte i B.



Strategi 2: (start i et andet punkt end B)
Vi starter ikke i B, dvs. vi kommer til byen ad en kant. Vi forlader B ad en anden kant, og derefter skal vi komme til B ad en tredje kant. Vi skal så forlade B ad den fjerde kant, og dermed slutter vi ikke i B.

Konklusion: Hvis B har graden 4 og vi starter i B, så slutter vi også i B. Omvendt gælder der, at hvis vi ikke starter i B, så slutter vi heller ikke i B.

Øvelse 13 Betragt en by B, hvortil der er seks veje.



a) Argumentér for, at vi får samme konklusion, som i tilfældet med fire veje (jf. eksempel 9).

b) Generalisér til situationen, hvor byen B har et lige antal veje.

Vi har i de ovenstående eksempler og øvelser set, at det er afgørende, om antallet af veje, der udgår fra en by, er et lige antal eller et ulige antal. Vi samler erfaringerne i følgende sætning:

Sætning 1

a) Hvis der fra en by i et vejnet udgår et ulige antal veje, skal vejvæsenet enten slutte eller starte i denne by, hvis de skal lave en rute, hvor alle kanter er med én gang.

b) Hvis der fra en by i et vejnet udgår et lige antal veje, så skal vejvæsenet slutte her, hvis de starter her. Omvendt så skal vejvæsenet ikke slutte her, hvis de ikke starter her.

På baggrund af sætning 1 kan vi argumentere for, at hvis et vejnet har mere end to byer med et ulige antal veje, så kan vi ikke finde en løsning til vejvæsenets problem. Hvis der fx er tre byer med et ulige antal veje, så vil der være en by med et ulige antal veje, hvor vejvæsenet hverken starter eller slutter, hvilket er i modstrid med resultatet i sætning 1.

Vi sammenfatter de vigtigste egenskaber ved Eulerture og Eulergrafer i den følgende sætning:

Sætning 2

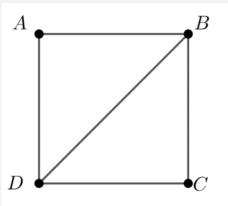
a) En graf er en Eulergraf, netop hvis alle hjørner har en lige grad.

b) Hvis der er højst to hjørner med en ulige grad, kan vi finde en Eulertur, men den er ikke lukket.

c) Hvis der er mindst tre hjørner med en ulige grad, er det ikke muligt at finde en Eulertur.

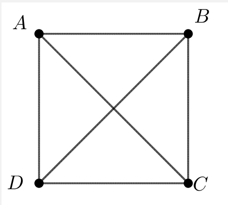
Vi beviser ikke sætningen her.

Eksempel 10 Figuren viser en graf, hvor der kan laves en Eulertur. Sætning 2b passer på grafen, da der er to hjørner med ulige grad, nemlig B og D.



Vandringen $D \rightarrow A \rightarrow B \rightarrow D \rightarrow C \rightarrow B$ er et eksempel på en Eulertur.

Figuren til højre viser næsten den samme graf, men der er tilføjet en kant. Nu kan der ikke længere findes en Eulertur, da alle fire hjørner har ulige grad, og vi kan derfor bruge sætning 2c.



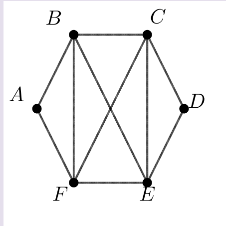
Øvelse 14 Find dit resultat af øvelse 10 om broerne i Königsberg. Findes der en løsning på Eulers problem, dvs. findes der en Eulertur?

Opgave 2

Figuren viser en graf.

a) Argumentér for, at grafen på figuren er en Eulergraf.

b) Angiv et eksempel på en lukket Eulertur.



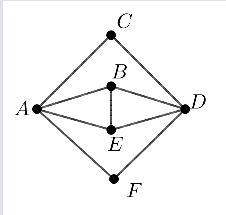
Opgave 3

Figuren viser en graf.

a) Argumentér for, at grafen på figuren ikke er en Eulergraf.

b) Argumentér for, at der findes en Eulertur på grafen.

c) Angiv et eksempel på en Eulertur på grafen.



Vi vender et øjeblik tilbage til det rekreative område fra øvelse 6, hvis billede er sat ind i opgaven herunder.

Opgave 4

Billedet viser et rekreativt område, hvor vejkrydsene er markeret med gule bogstaver.

a) Skitsér området i en graf.

b) Afgør, om man kan lave en Eulertur i området.

Udspændende træer

Vi introducerer nu et nyt begreb, der dækker over en ny slags delgrafer, der knytter sig til nogle andre problemstillinger end løberuter eller vandreture. Denne nye slags delgraf kan godt have forgreninger, hvilket en tur ikke har, da man ved forgreninger skal tilbage via mindst en af de kanter, hvor man allerede har vandret. De nye delgrafer bruges derfor netop til at undersøge, hvordan man skal håndtere ting, der kan fordele sig, fx strøm, vand og diverse signaler.

Definition 6 **Træ og udspændende træ**

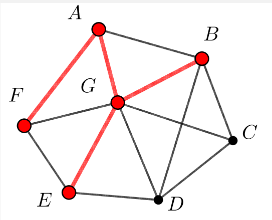
Et *træ* er en sammenhængende (del-)graf uden kredse.

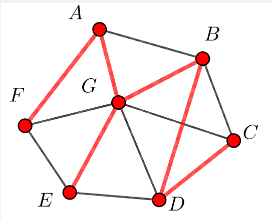
Et *udspændende træ* er et træ, hvor alle hjørnerne i grafen er en del af træet.

Eksempel 11 De to figurer herunder viser eksempler på en graf med markerede delgrafer.

På figuren til venstre er den markerede delgraf et træ, idet delgrafen er sammenhængende og uden kredse.

På figuren til højre er den markerede delgraf et udspændende træ, eftersom delgrafen er sammenhængende, uden kredse og indeholder alle hjørner.

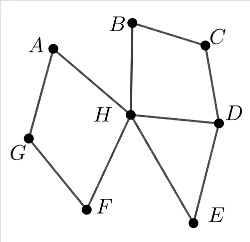




Øvelse 15

a) Marker et træ og et udspændende træ i grafen på figuren til højre.

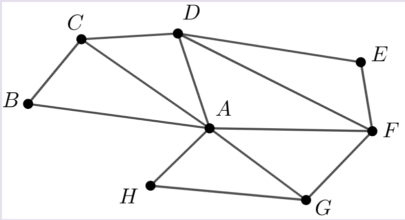
b) Hvor mange kanter er der i det udspændende træ, som du har markeret?



Opgave 5 Figuren viser en graf.

a) Markér et træ med tre hjørner i grafen.

b) Markér et udspændende træ i grafen.



Der er en simpel sammenhæng mellem antallet af hjørner i en graf og det antal kanter, der skal være i et udspændende træ i grafen:

Sætning 3 Et udspændende træ i en graf med H hjørner indeholder $H-1$ kanter.

I øvelse 16 og 17 undersøges, om sætning 3 er sand:

Øvelse 16

a) Tegn en graf med 1 hjørne, og overvej, at sætning 3 er sand, når $H=1$.

b) Tegn en graf med henholdsvis 2 og 3 hjørner, og overvej, at sætning 3 er sand, både når $H=2$, og når $H=3$.

Øvelse 17

a) Hvis sætning 3 er sand, når $H=n$, kan du så overbevise dig selv om, at den også er sand, når $H=n+1$?

b) Prøv f.eks. med $H=3$ og tilføj et ekstra hjørne til grafen; hvad sker der med det udspændende træ?

c) Generaliser til $H=n$.

Opgave 6

a) Tegn en graf med hjørnerne A, B, C, D, E, F, G og H, og med kanterne AB, AC, AD, BC, CD, CE, CH, DF, DH, EF, FG, FH og GH.

b) Marker et udspændende træ i grafen.

Tabeller og grafer

Nogle gange vil grafer være angivet ved en figur bestående af kanter og hjørner, sådan som vi har set det i de tidligere eksempler. Dette kan være med til at skabe et billedligt overblik. Andre gange vil en graf være repræsenteret ved en tabel.

Eksempel 12 Grafen i opgave 6 kan også repræsenteres i en tabel for overskuelighedens skyld. Et kryds markerer, at der er en kant mellem de to hjørner, hvorimod en streg markerer, at der ikke er en kant.

	A	B	C	D	E	F	G	H
A	-	×	×	×	-	-	-	-
B		-	×	-	-	-	-	-
C			-	×	×	-	-	×
D				-	-	×	-	×
E					-	×	-	-
F						-	×	×
G							-	×
H								-

Bemærk, at en fuldstændig udfyldt tabel ville være symmetrisk omkring diagonalen. Vi vælger derfor at undlade at udfylde den venstre nedre halvdel af tabellen.

Øvelse 18 Opstil en tabel, der beskriver grafen fra opgave 5.

Opgave 7 En venskabsgraf kan bruges til at beskrive, hvem der er venner med hvem i en gruppe. Hvert hjørne repræsenterer en person i gruppen, og to personer er venner, hvis der er en kant mellem deres hjørner.

Tegn en graf, der illustrerer venskabet i en gruppe på 5 personer ud fra nedenstående tabel, hvor ☺ angiver, at to personer er venner.

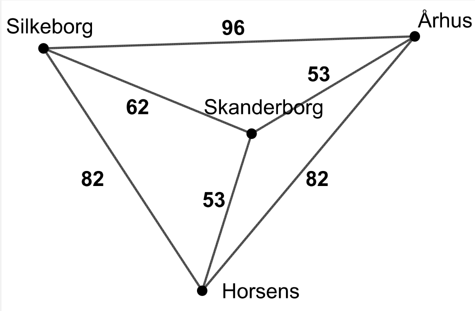
	Anton	Bolette	Camille	Daniel	Emma
Anton	-	☺	☺	☺	☺
Bolette		-	-	-	☺
Camille			-	☺	☺
Daniel				-	-
Emma					-

Vægtede grafer

Undertiden kan det være nyttigt at knytte tal til kanterne i en graf. Det kan fx være, hvis grafens hjørner betegner nogle byer, og kanternes tal angiver afstande mellem byerne, eller det kan angive transporttiden eller transportprisen. Grafer, hvor der er knyttet tal til kanterne, kaldes *vægtede* grafer.

Eksempel 13 Figuren viser en vægtet graf, der angiver prisen i kr. for en enkeltbillet med tog mellem de 4 byer i hjørnerne Silkeborg, Skanderborg, Århus og Horsens.

Grafen viser billetpriserne, men ikke de faktiske ruter, eftersom alle togene kører via Skanderborg.



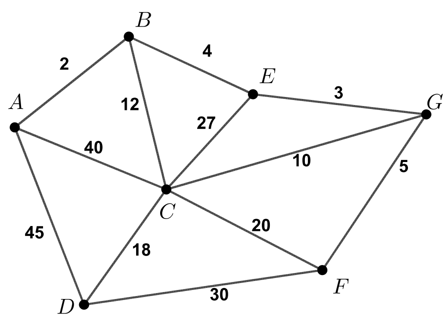
Eksempel 14 Den vægtede graf på figuren i eksempel 13 kan også repræsenteres ved en tabel:

	Horsens	Silkeborg	Skanderborg	Århus
Horsens	-	82	53	82
Silkeborg		-	62	96
Skanderborg			-	53
Århus				-

Prims algoritme

Lad os sige, at vi ønsker at sende signaler mellem byerne *A*, *B*, *C*, *D*, *E*, *F* og *G*, og at byerne derfor skal forbindes på en sådan måde, at det er muligt at sende signaler mellem alle byer, men uden at alle byer nødvendigvis er forbundet indbyrdes, dvs. signalet må gerne tage en omvej.

Vores teknikere har lavet vurderinger af, hvad prisen vil være for at forbinde nogle af byerne to og to, og resultatet anskues bedst ved den vægtede graf på figuren til højre, der har byerne som hjørner, og hvor kanternes vægte angiver prisen for at forbinde byerne.



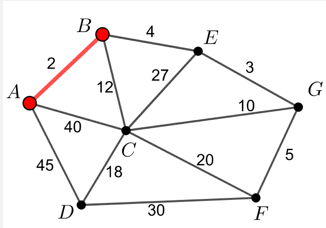
Pga. geografien er nogle strækninger dyrere end andre, og det er tydeligt, at det vil være en fordel, at have så få dyre strækninger med som muligt, hvis prisen skal holdes nede.

Prims algoritme gør det muligt for os at finde den billigste måde at lave et udspændende træ i grafen, og det er netop det, vi har brug for. Et udspændende træ når ud til alle hjørnerne, uden at man skal kunne bevæge sig ad en tur rundt til dem alle sammen; der må gerne være forgreninger. Man siger, at Prims algoritme finder det *minimale udspændende træ*, hvor ordet minimal fx kan dække over den mindste samlede pris eller den mindste afstand i hele træet.

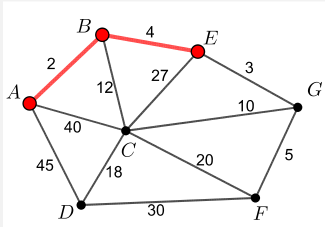
Nedenstående eksempel forklarer, hvordan algoritmen virker med udgangspunkt i grafen med byerne *A*-*F*.

Eksempel 15

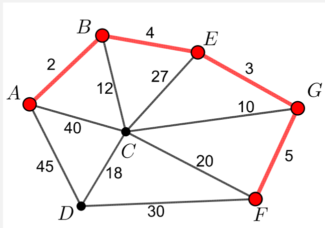
Vi starter fx i hjørne *A* og undersøger først, hvilken nabokant, det er billigst at tilføje. Det er kant *AB*, der koster 2, hvorimod *AC* og *AD* er dyrere.



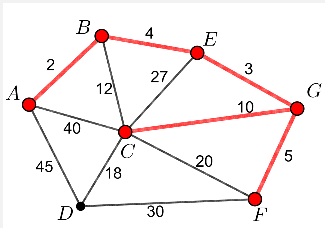
Så betragtes den nye delgraf *AB* og nabokanterne til delgrafen undersøges. Igen er *AD* og *AC* nabokanter, og det er *BC* og *BE* også. *BE* er den billigste, så hjørnet *E* tilføjes, og vores delgraf består nu af hjørnerne *A*, *B* og *E* med kanterne *AB* og *BE*.



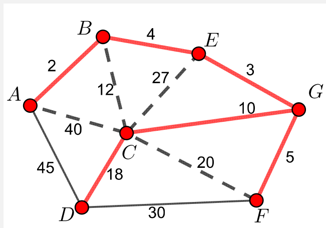
Nabokanterne til delgraf *ABE* er *AD*, *AC*, *BC*, *EC* og *EG*. Her er *EG* billigst, så hjørnet *G* tilføjes sammen med kanten *EG*, og på samme måde tilføjes *F* med kanten *GF* (se figuren).



Nu er der ganske mange nabokanter til delgrafen *ABEGF* nemlig *AD*, *AC*, *BC*, *EC*, *GC*, *FC* og *FD*. Den billigste af dem er *CG*, så den føjes til vores graf sammen med hjørne *C*.

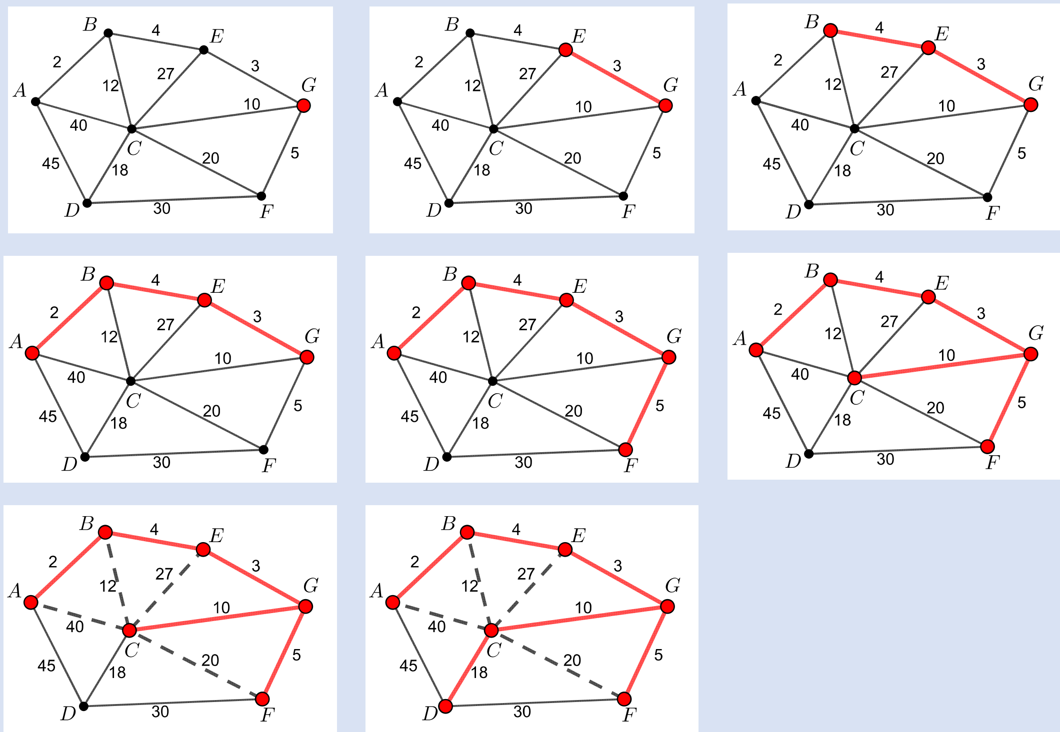


Igen er der mange nabokanter til vores delgraf: *AD*, *AC*, *BC*, *CE*, *CG*, *GF*, *DF*, *CD*. En del af dem giver anledning til kredse og fører til hjørner, der allerede er med i træet, så dem ser vi bort fra (se figuren, hvor de er markeret med stiplede linjer). Blandt de tilbageværende kanter *AD*, *DC* og *FD* er *DC* billigst, så den vælges, således at hjørnet *D* tilføjes som det sidste.



Prims algoritme fortæller derfor, at byerne skal forbindes som markeret med rødt på figuren, og den røde delgraf er det minimale udspændende træ i grafen. Den samlede pris bliver 2+4+3+5+10+18= 42.

Øvelse 19 Herunder ses Prims algoritme gennemført med udgangspunkt i hjørne G. Følg figurerne og gennemfør selv argumenterne for hver enkelt tilføjelse af en kant og et hjørne.



Øvelse 20 Gennemfør Prims algoritme i grafen med byerne fra eksempel 15 med udgangspunkt i hjørne D.

Øvelse 21 Tilføj en kant til det udspændende træ fra eksempel 15. Kan du gøre det uden at lave en kreds i delgrafen?

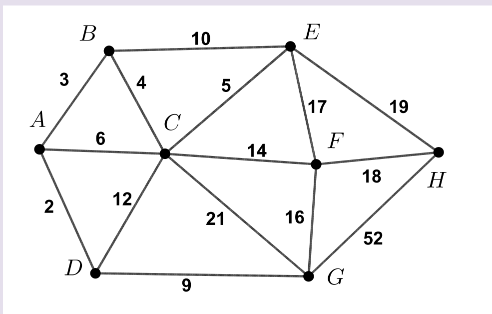
Ud fra de ovenstående øvelser kan vi se, at det minimale udspændende træ ser ud til at være uafhængigt af starthjørne. Det formulerer vi som en sætning:

Sætning 4 Det minimale udspændende træ, som man finder ved Prims algoritme, afhænger ikke af, hvilket hjørne i grafen algoritmen startes i.

Prims algoritme er en såkaldt "grådig" algoritme. Ordet "grådig" dækker over, at man i hvert enkelt trin i algoritmen skal gøre det, der virker mest optimalt i situationen, uden at være bekymret for, om det også er smartest på den lange bane. Det er det! Beviset ser vi på senere.

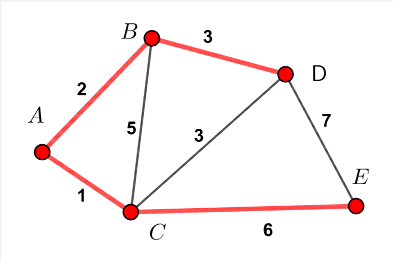
Opgave 8 Figuren viser en vægtet graf.

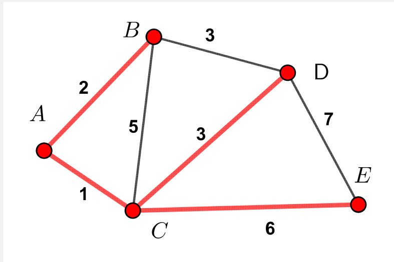
Gennemfør Prims algoritme, og bestem det minimale udspændende træ i grafen.



I ovenstående eksempler er alle kanternes vægte forskellige, men det kan godt forekomme, at nogle af vægtene er ens. Det betyder, at der godt kan være to eller flere forskellige minimale udspændende træer.

Eksempel 16 Figurerne nedenunder viser en graf med to mulige minimale udspændende træer. Kanterne *BD* og *CD* har begge vægten 3, og det giver anledning til to mulige resultater af Prims algoritme.



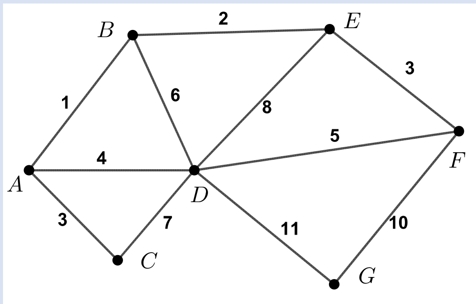


Bemærk her, at selvom der er flere minimale udspændende træer, så er de stadig uafhængige af starthjørnet for Prims algoritme, hvilket betyder, at sætning 4 også gælder, når grafen indeholder kanter med ens vægte. Man kan altså selv bestemme, hvilket hjørne man vil starte i.

Læg også mærke til, at der dannes en kreds, hvis der tilføjes en kant til det udspændende træ. Sammenlign med øvelse 21.

Øvelse 22 Figuren viser en graf, hvor kanterne *EF* og *AC* har samme vægt.

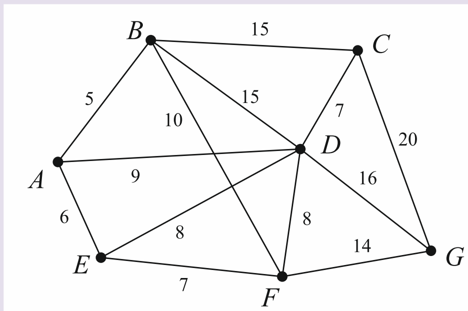
Find det/de minimale udspændende træer ved Prims algoritme.



Opgave 9

Figuren viser en graf med to par af kanter, der har samme vægt, nemlig parret *BC* og *BD* og parret *DE* og *DF*.

Gennemfør Prims algoritme, og find det/de minimale udspændende træer.



Øvelse 23 Opstil en tabel, der repræsenterer den vægtede graf i opgave 9.

Opgave 10 Tabellen viser en oversigt over en vægtet grafs kanter.

	A	*B*	*C*	*D*	*E*
A	-	-	3	7	5
B		-	4	-	2
C			-	6	8
D				-	-
E					-

a) Tegn en vægtet graf, der repræsenterer tabellen.

b) Gennemfør Prims algoritme, og find det minimale udspændende træ for grafen.

Indtil videre har vi stiltiende antaget, at Prims algoritme vil give os det minimale udspændende træ i en graf, men den påstand kræver naturligvis et bevis. Beviset er et *induktionsbevis*, hvor argumentet samtidig er et *modstridsargument*.

Induktionsbevis og modstrid

Induktionsbeviser bruges fx, hvis man skal vise, at en regel gælder for alle tal i en talfølge. Idéen er, at man skal vise, at **hvis** reglen gælder for ét tal i følgen, for eksempel tal nummer n, **så** gælder den også for det næste tal, dvs. tal nummer $n+1$. Dvs. at hvis vi kan vise, at reglen *altid gælder for det næste tal*, så får vi altså en dominoeffekt, der sikrer, at reglen gælder for alle tal, hvis man vel at mærke kan vise, at reglen gælder for bare ét tal ad gangen.

Processen i et induktionsbevis deles ofte op i to dele:
Induktionsstarten, hvor man viser, at reglen gælder for ét (eller flere) konkrete tal og *induktionstrinnet*, hvor man viser, at **hvis** sætningen gælder for tallet n, **så** gælder den også for tallet n+1.

Øvelse 24 Sammenlign ovenstående med øvelse 16 og 17. Kan du identificere induktionsstarten og induktionstrinnet?

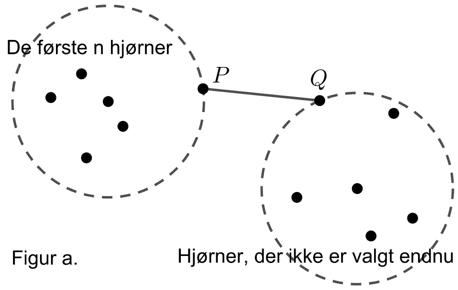
Når man anvender et modstridsargument til at vise en matematisk sætning, så starter man med at antage, at den modsatte påstand end sætningens er gældende. Vi kan kalde denne modsatte påstand for påstand A, som vi altså antager er sand. Man bruger derefter dette i sin videre argumentation, og når til sidst frem til noget, der strider mod ("er i modstrid med") enten vores andre antagelser i den matematiske sætning eller en anden gyldig matematisk sætning. Dermed kan man konkludere, at påstand A er falsk, og dermed må den modsatte påstand til påstand A være sand. I sidste ende betyder det, at den matematiske sætning er sand.

Bevis for Prims algoritme

Sætning 5 Prims algoritme finder det minimale udspændende træ i en graf.

Bevis:
Vi har vist sætningen, hvis vi kan vise, at når vi bruger Prims algoritme, så vil det træ, vi når frem til i hvert enkelt trin, være en del af det minimale udspændende træ. Vi ser for nemheds skyld på en graf, hvor alle kanter har forskellige vægte. Det minimale udspændende træ kaldes her for T. Da beviset er et induktionsbevis, indeholder det en induktionsstart og et induktionstrin.

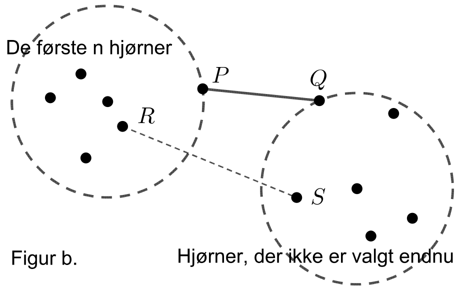
Induktionsstarten:
Vi starter i et tilfældigt hjørne, og det hjørne er naturligvis en del af *T*, eftersom *T* indeholder alle hjørner. Så vi ved altså, at vi har første hjørne i det minimale udspændende træ.



Figur a.

Induktionstrinnet:
Vi antager nu, at Prims algoritme har fundet *n* hjørner, og at det træ, der er fundet indtil videre, er en del af *T*. Vi skal vise, når vi bruger Prims algoritme til at tilføje endnu en kant og et hjørne, så der i alt er $n+1$ hjørner, så er resultatet stadig en del af *T*.

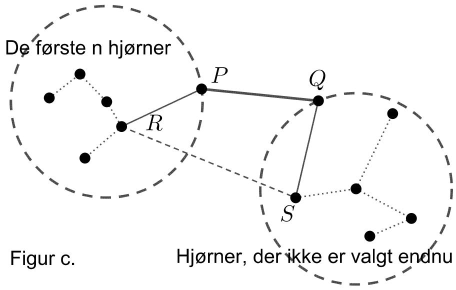
Vi gennemfører Prims algoritme og finder en kant, som vi her kalder *PQ*, hvor *P* er et hjørne i træet med de *n* hjørner, og *Q* er det hjørne, der blev tilføjet, se figur a. Vi skal argumentere for, at *PQ* er en del af *T*, og det gøres ved at opstille et modstridsargument, dvs. vi undersøger den modsatte påstand: "*PQ* er ikke med i *T"*.



Figur b.

Hvis *PQ* ikke er med i *T*, så må der være en anden kant i *T*, der forbinder de første *n* hjørner med de hjørner, der ikke er valgt endnu i vores proces. Lad os kalde den kant for *RS*, se figur b.

Men *RS* var jo én af de kanter, Prims algoritme kunne vælge mellem, da den valgte *PQ*, så vægten af *PQ* må være mindre end vægten af *RS*, da alle kanterne har forskellige vægte.



Figur c.

Hvis vi ser på den graf, der fremkommer ud fra *T* (dvs. med *RS*) sammen med kanten *PQ*, så vil *PQ* og *RS* indgå i en kreds i den graf, (se øvelse 21 og eksempel 16), og det betyder, at hvis vi fjerner *RS*, som jo var en del af det oprindelige udspændende træ *T,* og erstatter den med *PQ*, så har vi stadig et udspændende træ, og det er samtidig billigere end det oprindelige *T*. Se et eksempel på figur c.

Det betyder, at påstanden "*PQ* er ikke med i *T*", fører til den konklusion, at *T* ikke er det minimale udspændende træ. Men *T* **var** jo det minimale udspændende træ, det var en antagelse i sætningen, så her er den modstrid, der gerne skal opstå i modstridsbeviset. Vi kan derfor konkludere, at påstanden "*PQ* er ikke med i *T*" er falsk. Det modsatte må derfor gælde, og vi kan konkludere, at *PQ* er med i *T*.

Mere generelt kan vi konkludere, at når vi bruger Prims algoritme til at finde den kant, der skal forbinde de første *n* hjørner med hjørne nummer *n*+1, så vil den kant være en del af det minimale udspændende træ.

Nu har vi både induktionsstarten og induktionstrinnet, så vi ved, at når vi vælger det første hjørne, så er det med i *T*, og vi ved også, at hver gang vi gennemløber Prims algoritme, så finder vi en kant og tilføjer et hjørne, der begge er med i *T*.

Sætning 3 fortæller os, at hvis vi har *H* hjørner i en graf, så skal vi gennemløbe Prims algoritme $H-1$ gange, for så vil der være $H-1$ kanter og *H* hjørner i *T*, og det er netop det, der skal være i et udspændende træ.

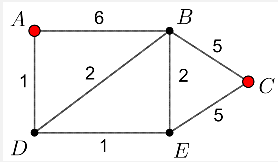
□

Dijkstras Algoritme

I en anden situation inden for grafteori ønsker man at finde den optimale sti mellem to hjørner i en vægtet graf. Ordet "optimal" skal forstås bredt. Det vil afhænge af, hvordan man vælger sine vægte, men målet er, at summen af vægtene skal være så lille som muligt på den sti, man tilbagelægger. I praksis betyder det, at man fx ønsker at finde den korteste, billigste eller hurtigste rute mellem to byer.

Det kan dog hurtigt blive en uoverskuelig opgave at holde regneskab med vægtene på de forskellige stier, hvis blot grafen har en vis størrelse. Man har derfor brug for en algoritme, som helt slavisk kan bestemme den optimale sti. *Dijkstras algoritme* er en metode, som netop kan bruges til at bestemme den optimale sti på en graf. Vi vil her se nærmere på, hvordan Dijkstras algoritme forløber.

Eksempel 17 **Dijkstras algoritme**



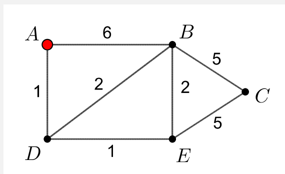
Vi ønsker at bestemme den korteste sti fra A til C i grafen til højre ved brug af Dijkstras algoritme. Dette eksempel viser, hvorledes algoritmen udføres trin for trin.

Hjørnerne kan tildeles to tilstande: *Midlertidig status* (M) og *Permanent status* (P). Ved starten af algoritmen har alle hjørnerne status M. Ét efter ét undersøges hjørnerne i henhold til algoritmen, hvorefter de overgår fra status M til status P. Når sluthjørnet har fået status P, stopper algoritmen, og den korteste sti fra starthjørnet (i dette tilfælde A) til sluthjørnet (i dette tilfælde C) er bestemt.

Undervejs registreres længden af den korteste (midlertidige) sti fra starthjørnet til et givet hjørne, og så længe der endnu ikke er angivet en sti til det pågældende hjørne, tildeles 'stien' længde ∞. Samtidigt registreres, hvilket hjørne der er besøgt umiddelbart før det pågældende hjørne. Dette hjælper os med at angive rækkefølgen af hjørner i den korteste sti fra starthjørnet til sluthjørnet. Resultatet af algoritmen opdateres løbende i en såkaldt *Dijkstra-tabel*.

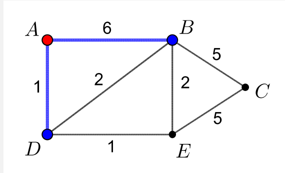
Trin 0.
Starthjørnet A undersøges. Da den korteste afstand fra A til A er naturligvis 0, indskrives dette i Dijkstra-tabellen. Alle øvrige hjørner tildeles en stilængde på ∞, eftersom længderne af mulige stier til de disse hjørner endnu ikke er kendte.

Hjørne	A	B	C	D	E
Korteste sti til A	0	∞	∞	∞	∞
Sidst besøgte hjørne					
Status	M	M	M	M	M



Trin 1.
Her vælges det hjørne med status M, som har den korteste sti til starthjørnet. I første runde er dette hjørnet A, der har en stilængde på 0 til starthjørnet. For dette aktuelle hjørne A undersøges nu den korteste afstand fra startpunktet til de af hjørnets nabohjørner, som har status M. Hjørnet A har to nabohjørner med status M, nemlig B og D (markeret med blåt i Dijkstra-tabellen nedenfor). Den korteste sti fra starthjørnet A til B er $0+6=6$. Da dette er en kortere sti end B's aktuelle sti på ∞, opdateres B's stilængde til 6. Tilsvarende bestemmes den korteste sti fra starthjørnet A til D, som er $0+1=1$, og D's stilængde opdateres til 1. Vi registrerer samtidigt, at det sidst besøgte hjørne på stien til hhv. B og D er A. Så er vi færdige med at undersøge hjørne A, der nu får status P.

Hjørne	*A*	*B*	*C*	*D*	*E*
Korteste sti til *A*	**0**	~~∞~~ 6	∞	~~∞~~ 1	∞
Sidst besøgte hjørne		*A*		*A*	
Status	~~*M*~~ ***P***	*M*	*M*	*M*	*M*

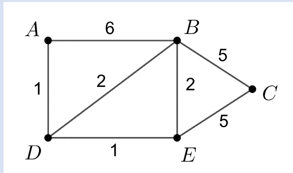


Trin 2.
Nu gentages trin 1, indtil sluthjørnet *C* har fået status *P*, hvorefter algoritmen stopper. Den korteste sti kan herefter aflæses i Dijkstra-tabellen. Bemærk, at alle hjørner ikke nødvendigvis har opnået status *P*, når algoritmen stopper.

Detaljerne i trin 2 fremgår af de tre næste øvelser (øvelse 25, 26 og 27) samt af eksempel 18, hvor det også fremgår, hvordan den korteste sti kan aflæses i Dijkstra-tabellen.

Øvelse 25 *Fortsættelse af eksempel* 17.
Tag udgangspunkt i Dijkstra-tabellen sidst i eksempel 17:

Hjørne	*A*	*B*	*C*	*D*	*E*
Korteste sti til *A*	0	6	∞	1	∞
Sidst besøgte hjørne		*A*		*A*	
Status	*P*	*M*	*M*	*M*	*M*



a) Begrund, at det aktuelle hjørne, som nu skal undersøges, er *D*.

b) Begrund, at de to nabohjørner til *D*, som vi nu vil kigge nærmere på, er *B* og *E*.

c) Bestem den korteste sti fra starthjørnet *A* til hvert af de to hjørner *B* og *E*.
Husk at stien skal gå gennem *D*.

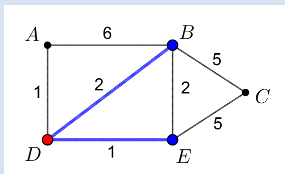
d) Opdater rækken med "korteste sti til *A*", hvis du fandt kortere stier fra hjørnerne *B* og *E* til starthjørne *A*.

e) Opdater rækken med "sidst besøgte hjørne", for hjørnerne *B* og *E*. (Vi fandt en kortere sti for både *B* og *E* til starthjørnet *A*, så derfor skal begge disse hjørner opdateres med *D* i rækken med "sidst besøgte hjørne").

f) Opdater rækken med "status", så hjørne *D* får status *P* (dvs. at hjørne *D* er færdigbehandlet).

g) Tjek at dine svar passer med nedenstående Dijkstra-tabel:

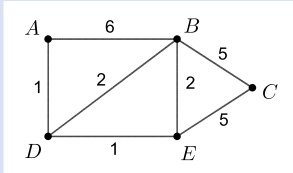
Hjørne	***A***	*B*	*C*	***D***	*E*
Korteste sti til *A*	**0**	~~6~~ 3	∞	**1**	~~∞~~ 2
Sidst besøgte hjørne		~~*A*~~ *D*		*A*	*D*
Status	***P***	*M*	*M*	~~*M*~~ ***P***	*M*



Øvelse 26 *Fortsættelse af øvelse* 25.

Gentag trin 1 i Dijkstras algoritmen endnu engang på grafen fra eksempel 17.
Tag udgangspunkt i den Dijkstra-tabel, som du fandt i øvelse 25:

Hjørne	***A***	*B*	*C*	***D***	*E*
Korteste sti til *A*	**0**	3	∞	**1**	2
Sidst besøgte hjørne		*D*		***A***	*D*
Status	***P***	*M*	*M*	***P***	*M*



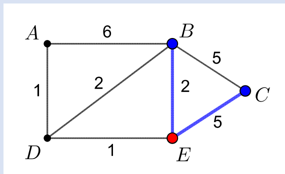
a) Hvilket hjørne med status *M* har den korteste sti til *A*? Dette hjørne skal nu undersøges.

b) Betragt grafen til højre. Hvilke nabohjørner med status *M* har det aktuelle hjørne, som du fandt i a)?

c) Bestem den korteste stilængde fra starthjørnet *A* til hvert af de to hjørner, som du fandt i b). Stierne skal begge gå gennem det hjørne, som du fandt i a).

d) Opdater Dijkstra-tabellen, og tjek, at du får nedenstående tabel:

Hjørne	***A***	*B*	*C*	***D***	***E***
Korteste sti til *A*	**0**	3	~~∞~~ 7	**1**	**2**
Sidst besøgte hjørne		*D*	*E*	***A***	***D***
Status	***P***	*M*	*M*	***P***	~~*M*~~ **P**



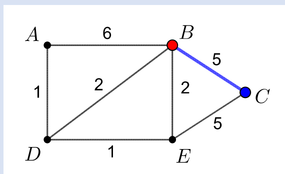
Øvelse 27 *Fortsættelse af øvelse* 26.

a) Gentag trin 1 i Dijkstras algoritme, og opdater Dijkstra-tabellen på ny.

b) Begrund, hvorfor stilængden for *C* ikke skal opdateres.

c) Tjek, at du når frem til følgende resultat:

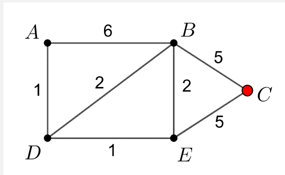
Hjørne	***A***	***B***	*C*	***D***	***E***
Korteste sti til *A*	**0**	**3**	7	**1**	**2**
Sidst besøgte hjørne		***D***	*E*	***A***	***D***
Status	***P***	~~*M*~~ ***P***	*M*	***P***	**P**



Eksempel 18 *Fortsættelse af eksempel* 17 (*samt af øvelse* 25, 26 og 27).

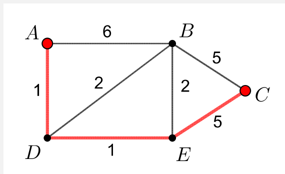
Trin 1 i Dijkstras algoritme gentages en sidste gang. Det eneste hjørne tilbage med status *M* er *C*. Og eftersom hjørnet *C* ikke har nabohjørner med status *M*, er der ikke nogen stier, som vi kan opdatere. Sluthjørnet *C* får derfor status *P*, og algoritmen stopper.

Hjørne	***A***	***B***	***C***	***D***	***E***
Korteste sti til *A*	**0**	**3**	**7**	**1**	**2**
Sidst besøgte hjørne		***D***	***E***	***A***	***D***
Status	***P***	***P***	~~*M*~~ ***P***	***P***	**P**



Resultatet opdateres i Dijkstra-tabellen, og det er nu muligt at aflæse den korteste sti fra starthjørnet *A* til sluthjørnet *C*. Faktisk kan vi aflæse den korteste sti fra *A* til ethvert hjørne, som har fået status *P*. I dette eksempel ender alle hjørner med status *P*, men det er ganske tilfældigt. Algoritmen slutter, når sluthjørnet opnår status *P*.

Hjørne	***A***	***B***	***C***	***D***	***E***
Korteste sti til *A*	**0**	**3**	**7**	**1**	**2**
Sidst besøgte hjørne		***D***	***E***	***A***	***D***
Status	***P***	***P***	***P***	***P***	**P**



Vi aflæser i Dijkstra-tabellen, at den korteste sti fra starthjørnet *A* til hjørnet *C* har en længde på 7. Rækkefølgen af hjørnerne i stien kan findes ved at læse Dijkstra-tabellen "baglæns": Vi kom til hjørnet *C* fra *E* – til hjørnet *E* fra *D* – og til hjørnet *D* fra *A*.

Vi kan nu konkludere:
Den korteste sti fra *A* til *C* er $A \to D \to E \to C$, og stien har en længde på 7.

Dijkstras algoritme til bestemmelse af den korteste sti fra et givet hjørne til grafens øvrige hjørner kan opsummeres på følgende måde:

Trin 0.
1. Tegn en Dijkstra-tabel, og tildel alle hjørner status *M*.
2. Tildel starthjørnet stilængde 0 og øvrige hjørner stilængde ∞.

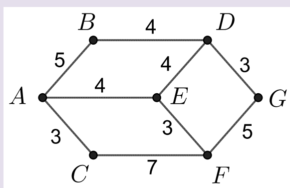
Trin 1.
3. Vælg det hjørne blandt hjørner med status *M*, der har den hidtil mindste stilængde til starthjørnet (i første runde vælges starthjørnet).
4. Bestem stilængderne fra starthjørnet via det valgte hjørne til dets nabohjørner med status *M*, og opdater stilængden i de tilfælde, hvor der kan opnås en kortere sti.
5. Hvis stilængden opdateres, så angiv hvilket hjørne, der sidst er besøgt.
6. Giv det valgte hjørne i punkt 3. status *P*, når det er blevet færdigbehandlet.

Trin 2.
7. Gentag punkterne under trin 1, indtil sluthjørnet opnår status *P*.
8. Ved at læse Dijkstra-tabellen "baglæns" kan den korteste sti fra starthjørnet til sluthjørnet bestemmes.

Opgave 11 På figuren til højre ses en vægtet graf.

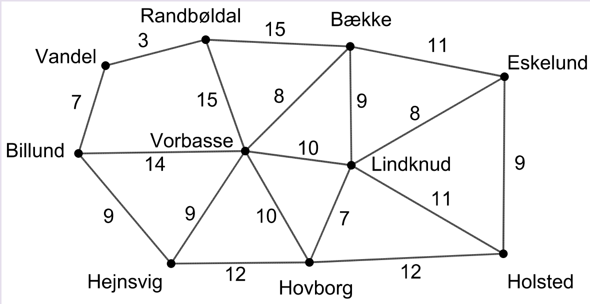
a) Opstil ved brug af Dijkstra algoritme en Dijkstra-tabel, som angiver længden af den korteste sti fra hjørnet *A* til hvert af de øvrige hjørner i grafen.

b) Angiv rækkefølgen af punkter på den korteste sti fra hjørne *A* til hjørne *G*.



Opgave 12

Grafen til højre angiver afstande (målt i km) mellem en række byer i Jylland. Man ønsker en oversigt over de korteste ruter fra Holsted til hver af de øvrige ni byer.



a) Anvend Dijkstras algoritme til at udfylde Dijkstra-tabellen nedenfor, så alle byer opnår status *P*.

b) Anvend Dijkstras algoritme til at bestemme den korteste rute fra Holsted til Vandel.

	Billund	Bække	Eskelund	Hejnsvig	Holsted	Hovborg	Lindknud	Randbøldal	Vorbasse	Vandel
Korteste vej til Holsted										
Sidst besøgte by										
Status										

Indstik til formelsamlingen

Dijkstras algoritme til bestemmelse af den korteste sti fra et givet hjørne til grafens øvrige hjørner

Trin 0.

1. Tegn en Dijkstra-tabel, og tildel alle hjørner status M.
2. Tildel starthjørnet stilængde 0 og øvrige hjørner stilængde ∞.

Trin 1.

3. Vælg det hjørne blandt hjørner med status M, der har den hidtil mindste stilængde til starthjørnet (i første runde vælges starthjørnet).
4. Bestem stilængderne fra starthjørnet via det valgte hjørne til dets nabohjørner med status M, og opdater stilængden i de tilfælde, hvor der kan opnås en kortere sti.
5. Hvis stilængden opdateres, så angiv hvilket hjørne, der sidst er besøgt.
6. Giv det valgte hjørne i punkt 3. status P, når det er blevet færdigbehandlet.

Trin 2.

7. Gentag punkterne under trin 1, indtil sluthjørnet opnår status P.
8. Ved at læse Dijkstra-tabellen ”baglæns” kan den korteste sti fra starthjørnet til sluthjørnet bestemmes.

Dijkstra-tabel med n hjørner og starthjørne H_i

Hjørne	H_1	H_2	H_3	$\cdots$	H_{n-1}	H_n
Korteste sti til H_i						
Sidst besøgte hjørne						
Status						

Prims algoritme til bestemmelse af det minimale udspændende træ

1. Vælg et hjørne i grafen.
2. Vælg den billigste kant, der ligger ved hjørnet, og tilføj kanten og hjørnet, som kanten fører hen til.
3. Vælg igen den billigste kant, der ligger ved ét at de valgte hjørner, og tilføj kanten og det hjørne, som kanten førte hen til.
4. Gentag proceduren i 3. under forudsætning af, at du ikke laver en kreds i den valgte delgraf.
5. Slut proceduren, når du har alle hjørner med.