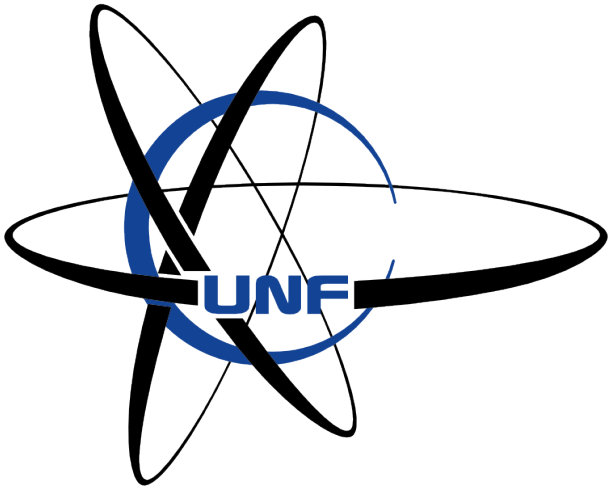
Workshop i grafteori

Rasmus Frigaard Lemvig - rle@unf.dk

Ungdommens Naturvidenskabelige Forening, København

Dato: 07-12-2021



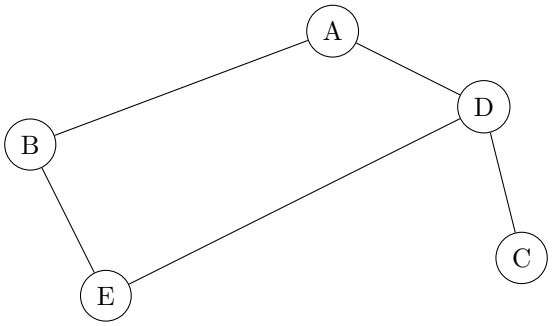
Introduktion til workshoppen

I denne workshop skal vi arbejde med grafteori, som er en gren af såkaldt "diskret matematik", som man kan tænke på som "matematik uden kommatal". Workshoppen består af tre dele: en introduktion, hvor vi får de grundlæggende begreber under huden, et kapitel om Eulerture, der er vigtige for grafteoriens historie, og til slut arbejder vi kort med vægtede grafer. De primære formål med workshoppen er, at I bliver klogere og oplever matematik på en anden måde, end man gør i grundskolen og gymnasiet. Opgaverne i dette kompendium er inddelt i tre sværhedsgrader, let, mellem og svær indikeret med hhv. 1, 2 og 3 prikker.

1 Hvad er grafteori?

1.1 Vigtige begreber

Grafteori er studiet af grafer. Du tænker måske på grafer af funktioner i et koordinatsystem, men dette er en helt anden type graf. En graf i denne workshop er en samling af knuder og kanter, der forbinder knuderne med hinanden. Lad os se på et eksempel:



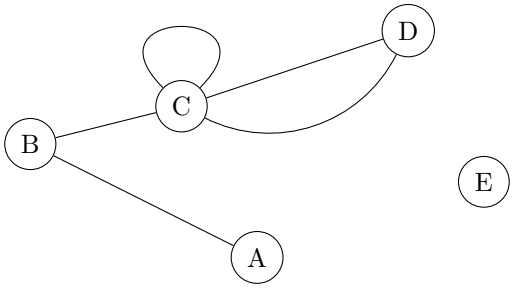
Figur 1: Eksempel på en graf

Knuder tegner vi som cirkler eller firkanter, som kan have et navn (her blot bogstaver), mens kanterne er linjerne, der forbinder knuderne. Man kan tænke på grafer på flere måder. Forestil dig f.eks. et vejkort. Her er knuderne placeringer og kanterne veje. I dette afsnit skal vi lære de mest grundlæggende begreber i grafteori, så vi er klar til at løse rigtige problemer i de senere afsnit. Der kan være en del begreber at huske, men heldigvis er de fleste af navnene ret intuitive.

Definition 1.1. Vi har følgende definitioner:

1. En *løkke* er en kant, som starter og slutter i samme knude.
2. To forskellige knuder kaldes *naboknuder*, hvis der er en kant, der forbinder dem.
3. *Valensen* af en knude er lig antal kanter, der har endepunkt i knuden (en løkke lægger 2 til valensen). Den *totale valens* for en graf er summen af valensen for alle knuder i grafen.
4. En *isoleret* knude er en knude, hvor ingen kanter har endepunkt. Med andre ord, en knude med valens 0.

Lad os illustrere disse begreber med en figur:



Figur 2: Illustration af begreberne i definition 1.1. A og B er et eksempel på to naboknuder. C har en løkke. E er en isoleret knude, idet denne knude ingen kanter har. Valensen af A er 1, for B er den 2. Valensen af C og D er henholdsvis 5 og 2.

Når man i matematik har indført en definition, vil man ofte gerne bygge ovenpå definitionerne med nogle resultater. Valens er et vigtigt begreb, som vi kommer til at bruge en del fremover. Vi har denne sætning om valensen i en graf:

Sætning 1.2. *Den totale valens for en graf er lig antallet af kanter gange 2.*

Bevis. Alle kanter i en graf starter og slutter i en knude (muligvis den samme knude). Altså må alle kanter bidrage med 2 til grafens totale valens. ■

Når man har vist en sætning i matematik, kan det ske, at man får et nyttigt resultat eller to mere herudfra. Sådan en følgesætning kaldes et *korollar*. Vi har sådan en hjælpesætning:

Korollar 1.3. *Den totale valens for en graf er et lige tal.*

Bevis. Da den totale valens for en graf er lig 2 gange antallet af kanter, må 2 gå op i den totale valens. Altså er den totale valens lige. ■

Når vi arbejder med grafer, vil vi gerne kunne beskrive, hvordan forskellige knuder er relateret til hinanden i grafen. Derfor indfører vi nogle flere begreber:

Definition 1.4. Lad G være en graf med knuder A og B.

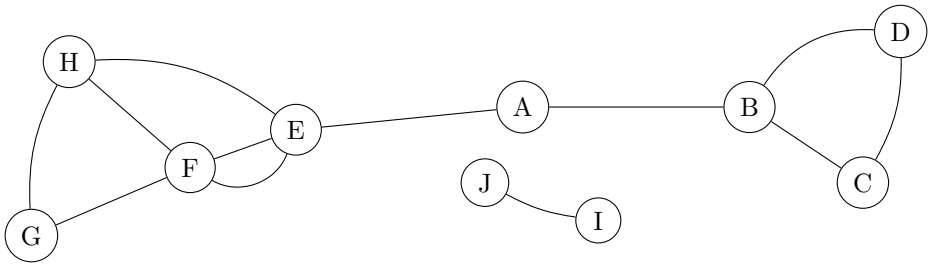
1. Der er en *rute* mellem A og B, hvis vi kan komme fra A til B ved at bevæge os langs nogle knuder og kanter i grafen. Følgen af knuder og kanter, vi bevæger os igennem, er selve ruten.
2. En *tur* fra A til B er en rute fra A til B, hvor man kun må gå langs hver kant én gang.
3. En *vej* fra A til B er en tur fra A til B, hvor man kun må passere hver knude én gang.
4. En rute kaldes *lukket*, hvis første og sidste knude på ruten er ens. Samme gælder for en lukket tur og en lukket vej. Hvis en rute, tur eller vej ikke er lukket, kaldes den *åben*.
5. En *kreds* er en lukket tur, hvor de eneste gentagne knuder er start- og slutknuden.

For at gøre definitionerne lidt lettere at huske, har vi følgende tabel:

Table 1: Tabel til begreberne i definition 1.4

Definition	Gentagne kanter	Gentagne knuder	Samme start- og slutknude
Rute	Tilladt	Tilladt	Tilladt
Tur	Nej	Tilladt	Tilladt
Vej	Nej	Nej	Nej
Lukket rute	Tilladt	Tilladt	Ja
Lukket tur	Nej	Tilladt	Ja
Kreds	Nej	Kun første og sidste	Ja

Ser vi på figur 2, kan vi se, at der er ruter mellem alle knuder undtagen E. Der findes også nogle lukkede ture. F.eks. kan vi gå fra D til C langs én af kanterne og tilbage til D gennem den anden kant. Denne tur er også en kreds. Kan du finde andre lukkede ture? Andre kredse? En anden illustration af begreberne ses i grafen herunder:



Figur 3: En graf med mange slags ruter.

I denne graf er der mange ruter! Dog er der ikke ruter mellem alle knuder, f.eks. findes ingen rute mellem I og A. Vi kan ligeledes finde mange ture, også lukkede af slagsen. F.eks. kan vi gå fra B til C til D og tilbage til B. Dette er også en kreds. Kan vi finde en lukket tur, som ikke er en kreds? Ja, lad os starte i F og gå til G, til H, tilbage til F, til E og tilbage til F langs den anden kant. Dette er en lukket tur, da ingen kanter bliver brugt mere end én gang, og vi både starter og slutter i F. Men vi passerer F undervejs i turen, så F optræder ikke kun som start- og slutknude. Derfor er denne lukkede tur ikke en kreds.

Se igen på figur 3. Der er noget særligt ved knuderne I og J i forhold til de andre knuder i grafen. I og J kan forbindes med en rute, men de kan ikke forbindes med ruter til nogle andre knuder i grafen. På samme måde kan knuderne fra A til H forbindes med ruter indbyrdes. Disse overvejelser giver os ideen til en ny definition:

Definition 1.5. En graf kaldes *sammenhængende*, hvis det for alle par af knuder i grafen gælder, at disse kan forbindes med en rute.

Graferne i figur 2 og 3 er ikke sammenhængende. Det er grafen på figur 1 til gengæld.

1.2 Nogle graftyper

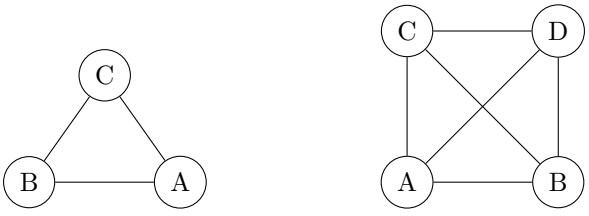
Vi vil nu komme ind på nogle bestemte typer grafer, som er gode at kende.

Definition 1.6. En *simpel graf* er en graf uden løkker, og hvor ingen naboknuder er forbundet af mere end én kant.

Grafen i figur 1 er et eksempel på en simpel graf, mens grafen i figur 2 ikke er en simpel graf. F.eks. har knude C en løkke. Simple grafer bliver vigtige i sidste afsnit, når vi ser på vægtede grafer. Man kan tænke på simple grafer som grafer uden "overflødige kanter", altså der er netop det antal kanter i grafen, som skal bruges, for at de samme knuder er forbundet. Vi ser nu på en speciel type af simple grafer.

Definition 1.7. En *komplet graf* er en simpel graf, hvori alle par af forskellige knuder er forbundet med hinanden. Den komplette graf med n knuder betegner vi som K_n.

Lad os se på de to komplette grafer K_3 og K_4 som eksempel:

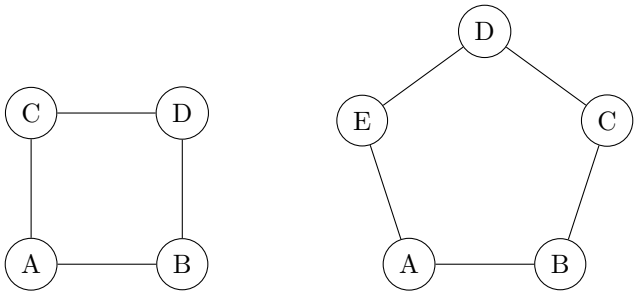


Figur 4: Graferne K_3 og K_4.

Den sidste type graf, vi vil kigge på her, er kredsgrafer. Definitionen er meget, som navnet antyder.

Definition 1.8. En *kredsgraf* er en simpel graf, hvor alle knuderne indgår i netop én kreds. Kredsgrafen med $n \geq 3$ knuder betegner vi som C_n (C'et står for "cycle", det engelske ord for "kreds").

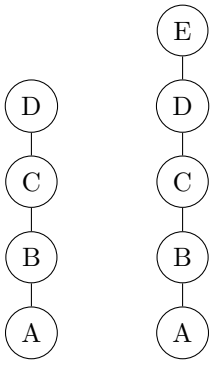
Hvorfor skal vi have $n \geq 3$? Hvis n er 2, kan vi slet ikke lave en kreds uden at skulle tegne flere kanter mellem de to knuder. Men så kan grafen ikke være simpel! Definitionen giver altså kun mening, hvis $n \geq 3$. Lad os se på et eksempel, nemlig graferne C_4 og C_5:



Figur 5: Graferne C_4 og C_5.

Definition 1.9. En *vejgraf* er en simpel graf, hvor én vej kan gennemløbe alle knuder og kanter i grafen. Vejgrafen med n knuder betegner vi P_n (P'et står for "path", det engelske ord for "vej").

Nedenfor ser vi to eksempler på vejgrafer.



Figur 6: Graferne P_4 og P_5.

Vi kan afslutte med at bemærke, at komplette grafer, kredsgrafer og vejgrafer (de tre standardgrafer) altid er sammenhængende. Vi er nu parate til at kaste os ud i at løse nogle interessante problemer med grafteori.

1.3 Opgaver til Hvad er grafteori?

- **Opgave 1.1:**
  Tegn følgende:

  1)En graf med tre knuder med valens 1, 2 og 3.

  2)En graf med fire knuder alle med valens 1.

  3)En graf med fem knuder med valens 1, 1, 2, 2 og 4.

  4)En ikke-sammenhængende graf med fire knuder, alle med valens 2.

- **Opgave 1.2:**
  Tegn dit hus/din lejlighed som en graf, hvor et værelse er en knude, og kanterne vejene mellem disse. Hvordan kan man f.eks. håndtere flere etager?

- **Opgave 1.3:**
  Tegn en sammenhængende graf uden løkker, der indeholder netop 2 kredse.

- **Opgave 1.4:**
  Hvor mange grafer kan du tegne, som er sammenhængende, simple, og hvor alle knuder har lige valens? Kender du nogle i forvejen? Hvor mange findes der?

- **Opgave 1.5:**
  Tegn graferne:

  1)Den komplette graf med 5 knuder, K_5.

  2)Kredsgrafen med 6 knuder, C_6.

  3)En sammenhængende graf med 7 knuder. Knuderne skal have valens 1, 2, 3, 4, 5, 6 og 7.

- **Opgave 1.6:**
  Tegn en graf med 4 knuder. Knuderne skal have valens 1, 2, 2 og 3. Findes en graf med fire knuder af valens 1, 2, 2 og 4? Hvis ja, tegn den. Hvis nej, hvorfor ikke?

- **Opgave 1.7:**
  Se på K_n og C_n. For hvilke værdier af n er disse grafer ens? Husk, at $n \geq 3$.

- **Opgave 1.8:**
  I disse opgaver skal vi se på, om det er muligt i en gruppe af personer, at hver person er ven med et bestemt antal personer i gruppen. Vi antager, at hvis person x er ven med person y, er person y også ven med person x. Hvis det er muligt, tegn da en graf, som illustrerer vennerelationerne.

  1)Antag, at gruppen er på 4 personer. Er det muligt for alle gruppemedlemmer at være venner med præcist 2 andre i gruppen?

  2)Antag, at gruppen er på 7 personer. Er det muligt, at alle gruppemedlemmer er venner med præcist 5 andre i gruppen?

  3)Antag, at gruppen er på 211 personer. Er det muligt, at alle gruppemedlemmer er venner med præcist 193 andre i gruppen?

- **Opgave 1.9:**
  Bevis, at der i alle grafer er et lige antal knuder med ulige valens. [Vink: brug korollar 1.3]

●● **Opgave 1.10:**
Giv et argument for, at alle sammenhængende grafer med n knuder har mindst $n-1$ kanter.

●●● **Opgave 1.11:**

1)Vis, at K_n har $\frac{n(n-1)}{2}$ kanter [Vink: Tegn K_n for f.eks. $n = 5$ og tæl op for hver kant, brug dette til at give et argument for det generelle tilfælde].

2)Vis, at en simpel graf med n knuder højst kan have $\frac{n(n-1)}{2}$ kanter.

3)Find en simpel graf, der har dobbelt så mange kanter, som den har knuder. [Vink: brug første delopgave]

De følgende opgaver omhandler en bestemt type graf, nemlig et *træ*.

Definition 1.10. Et *træ* er en sammenhængende graf, der ikke indeholder nogle kredse.

●●● **Opgave 1.12:**
I denne opgave viser vi nogle generelle egenskaber for træer.

1)Tegn et træ med henholdsvis 3, 5 og 10 knuder.

2)Bevis, at et træ bliver nødt til at være simpelt.

3)Hvor mange træer findes der med 4 knuder? Hvad med 5 knuder?

4)Findes et træ med 7 knuder og 7 kanter? Hvis ja, tegn træet. Hvis nej, hvorfor?

5)Hvor mange kanter er i et træ med n knuder? [Vink: brug opgave 1.10. Hvad sker der, hvis man tilføjer flere kanter?]

6)Hvad er den totale valens for et træ med n knuder?

●●● **Opgave 1.13:**

Bevis følgende resultat:

Proposition. *Lad G være en graf med n knuder. Hvis G er sammenhængende og har $n-1$ kanter, er G et træ.*

[Vink: Antag, at G indeholder en kreds. Vis, at dette fører til en selvmodsigelse, og konkludér, at G ikke kan have en kreds, ergo må G være et træ]

Definition 1.11. Et *blad* er en knude i en graf med valens 1.

●●● **Opgave 1.14:**

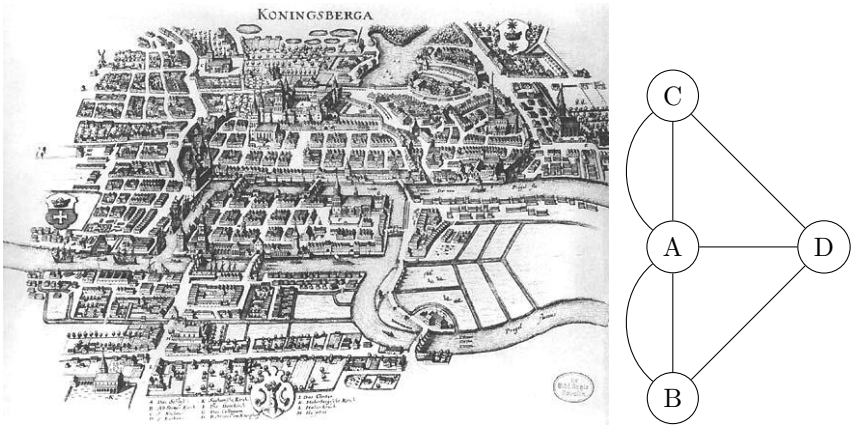
1)Bevis, at der findes mindst ét blad i et træ. [Vink: start i en vilkårlig knude og hop videre til den næste]

2)Bevis, at der findes mindst to blade i et træ. [Vink: brug 1)]

2 Eulerture

2.1 Grafteoriens fødsel - historien om Königsberg

I 1736 blev Leonhard Euler bedt om at finde en rute gennem byen Königsberg, så en procession (et optog) kunne gå over samtlige 7 broer i byen præcis én gang (som I måske husker fra tidligere, er det definitionen af en "tur"). Året efter udgav han en artikel, som analyserede, hvorvidt dette var muligt. Denne artikel betragtes som det første bidrag til grateorien [2]. Nedenfor på figur 7 ses et billede af Königsbergs broer, og hvordan dette kan repræsenteres med en graf.



Figur 7: *Venstre*: Tegning af Königsberg med broerne [3]. *Højre*: Königsberg repræsenteret ved en graf. Knuderne er de fire landområder, og kanterne er broerne mellem dem.

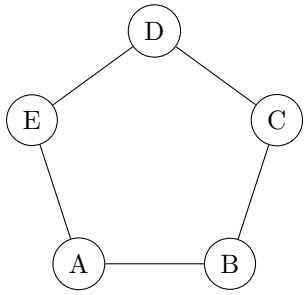
2.2 Åbne og lukkede Eulerture

Efter dette store bidrag til grafteorien har Euler naturligvis fået et begreb opkaldt efter sig:

Definition 2.1. En *Euler-tur* er en tur, som indeholder alle grafens kanter.
En Euler-tur kan enten være åben eller lukket. Her genbruger vi blot definition 1.4-(4):

- En lukket Euler-tur starter og slutter i samme knude.
- En åben Euler-tur starter og slutter i to forskellige knuder.

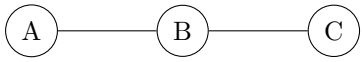
Eksempel 2.2. For et helt simpelt eksempel på en lukket Euler-tur kan vi se på kredsgrafen med 5 knuder igen:



For at lave en lukket Euler-tur kan vi vælge en vilkårlig knude at starte i. For eksempel kan vi starte i A. Vi kan nu gå turen $A \to B \to C \to D \to E \to A$. Da alle kanterne er med i turen netop én gang og vi har samme start- og slutknude, har vi altså lavet en lukket Euler-tur i grafen!

Overvej: Gælder det for alle kredsgrafer?

Eksempel 2.3. For et helt simpelt eksempel på en åben Euler-tur kan vi se på vejgrafen med 3 knuder P_3:



For at lave en åben Euler-tur skal vi vælge en knude at starte i med en smule omtanke. Vi skal nemlig starte og ende i en af endeknuderne for, at vi kan få alle kanter og knuder med. For eksempel kan vi starte i A. Vi kan nu gå turen $A \to B \to C$. Da alle kanterne er med i turen netop én gang, og vi har forskellige start- og slutknuder, har vi altså lavet en åben Euler-tur i grafen!

Nu har vi kun kigget på simple eksempler, men det er ikke svært at forestille sig meget store grafer med hundredevis af knuder og tusinder af kanter imellem sig. Hvordan kan vi finde ud af, om den har en Eulertur? Det siger denne sætning noget om:

Sætning 2.4. *Betragt en graf G uden isolerede knuder.*

1. *G har en lukket Eulertur, hvis og kun hvis den er sammenhængende og alle knuder har lige valens.*
2. *G har en åben Eulertur, hvis og kun hvis den er sammenhængende og har præcist to knuder med ulige valens.*

Bevis. For at bevise påstand 1, skal vi vise to ting. Første skal vi vise, at hvis G har en lukket Eulertur, er G sammenhængende, og alle knuder har lige valens. Dette overlades som en øvelse, se opgaverne. Dernæst skal vi vise, at hvis en graf G er sammenhængende, og alle knuder har lige valens, vil G have en lukket Eulertur. Denne implikation udelades, da den bygger på et induktionsargument, som rækker en del udover workshoppens indhold. At bevise påstand 2 forløber på tilsvarende vis. ■

2.3 Opgaver til Eulerture

Vi konkluderede i eksempel 2.2, at alle kredsgrafer har lukkede Eulerture. I de næste to opgaver ser vi på de to andre typer af standardgrafer K_n (komplet graf) og P_n (vejgraf) og undersøger, om disse har Eulerture (åbne eller lukkede).

- **Opgave 2.1:**
  [Vink: Tegn graferne!]

  1)Har P_4 en Eulertur? Hvis ja, er den så åben eller lukket?

  2)Har en vilkårlig vejgraf P_n en Euler-tur? Hvis ja, er den så åben eller lukket?

- **Opgave 2.2:**
  [Vink: Tegn graferne!]

  1)Har K_4 en Eulertur? Hvis ja, er den så åben eller lukket?

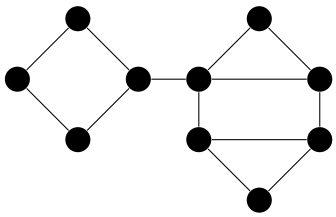
  2)Har K_5 en Eulertur? Hvis ja, er den så åben eller lukket?

  3)Har K_6 en Eulertur? Hvis ja, er den så åben eller lukket?

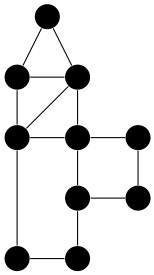
  4)Hvilke komplette grafer har en Eulertur? Hvad skal der gælder om n?

- **Opgave 2.3:**
  Denne opgave er fra [2]. Afgør om graferne har en lukket eller åben Eulertur, og find om muligt en.

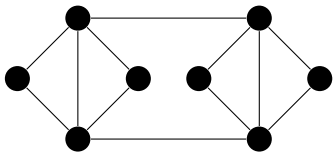
  1)



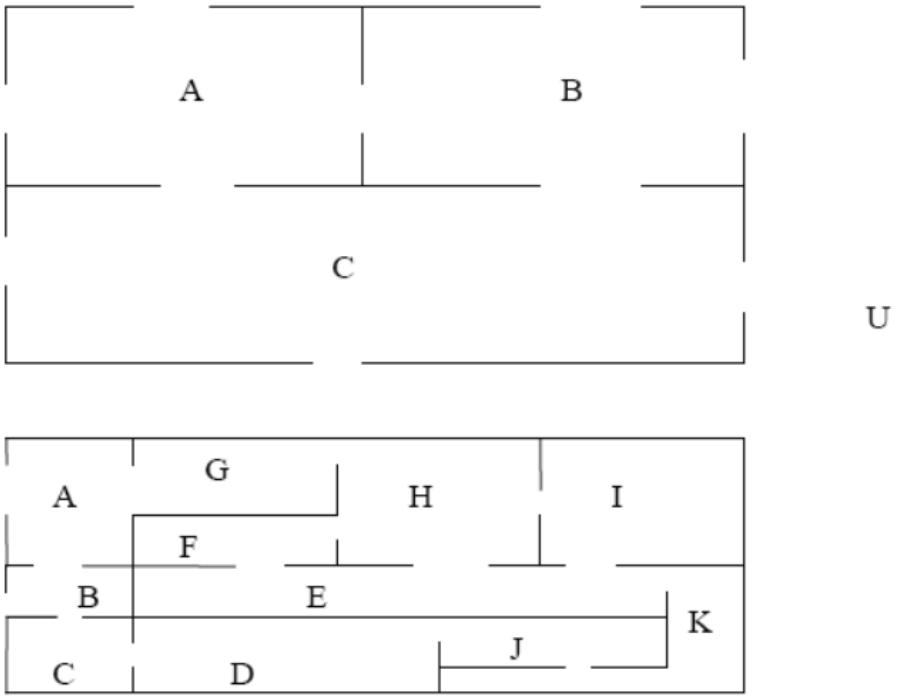
  2)



  3)



●● **Opgave 2.4:**
Denne opgave er fra [2]. Herunder er tegnet en plan over to huse. Tegn passende grafer over begge huse, så rummene er knuder og dørene kanter (betragt haven U som et rum i hvert hus). Afgør herefter, om det er muligt at gå en tur gennem huset, så man går gennem hver dør netop én gang.



Figur 8: Plan over de to huse.

●● **Opgave 2.5:**
Kan man finde en lukket Eulertur over broerne i Königberg?

●● **Opgave 2.6:**
Kan man finde en åben Eulertur over broerne i Königberg?

●●● **Opgave 2.7:**
Lad G være en graf med en lukket Eulertur.

1)Bevis, at G er sammenhængende.

2)Bevis, at alle knuder i G har lige valens.

●●● **Opgave 2.8:**
Lad G være en graf med en åben Eulertur.

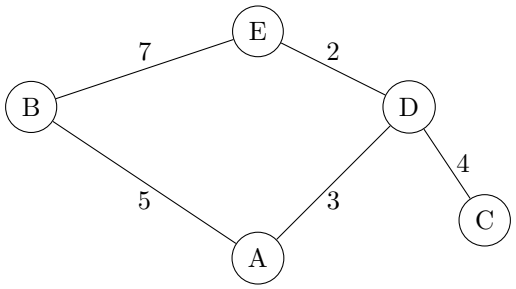
1)Bevis, at G er sammenhængende.

2)Bevis, at netop to knuder i G har ulige valens. [Vink: hvad skal gælde om valensen af de knuder, som turen starter og slutter i? Hvad med alle andre knuder?]

3 Vægtede grafer og grafalgoritmer

3.1 Vægtede grafer

En vægtet graf er blot en graf, hvor alle kanter har en værdi tilknyttet. Denne værdi kaldes en vægt. Et eksempel ses herunder:



Figur 9: En simpel vægtet graf

Man kan fortolke vægtede grafer som mange ting. Et eksempel kunne være steder og veje mellem disse. Vægtene kan her være afstande, omkostninger med mere. I virkeligheden vil man ofte være interesseret i at finde den korteste vej fra et sted til et andet. I grafteori kan dette omsættes til: Givet en vægtet graf G, find den rute fra knude A til knude B, hvor summen af vægten af kanterne i ruten er mindst mulig. Det er netop dette problem, som Djikstras algoritme løser (Djikstra udtales "deigstra"), hvis man lader alle vægtene være positive.

Lad os overveje en anden fortolkning af vægtede grafer. Forestil dig, at du har et netværk af f.eks. fabrikker, og du ønsker at bygge et netværk af veje mellem dem. Det skal være muligt at komme fra en vilkårlig fabrik til en hvilken som helst anden fabrik i netværket, men du ønsker ikke at lave så meget som én vej for meget. Vi kan lade samtlige muligheder for veje udgøre en sammenhængende graf G. Problemet svarer da til at finde et træ, der udspænder grafen, dvs. et træ, som indeholder de samme knuder som grafen. Sådan et træ kaldes et *udspændende træ*. Hvis G er en vægtet graf, og vi ydermere ønsker, at vægten af træet (dvs. summen af vægtene for alle kanterne i træet) skal være mindst muligt, kaldes det et *mindste udspændende træ*. Kruskals algoritme er én af mange algoritmer, der finder et mindste udspændende træ i en vægtet graf.

3.2 Djikstras algoritme og korteste veje

Nu gennemgås Djikstras (udtales *deigstra*) algoritme. For at algoritmen kan begynde, skal den have en startknude og en vægtet graf, hvor alle vægtene (kanternes værdier) er positive. Djikstras algoritme findes i flere udgaver. Den udgave, vi tager udgangspunkt i, giver den korteste afstand fra startknuden til alle andre knuder i grafen.

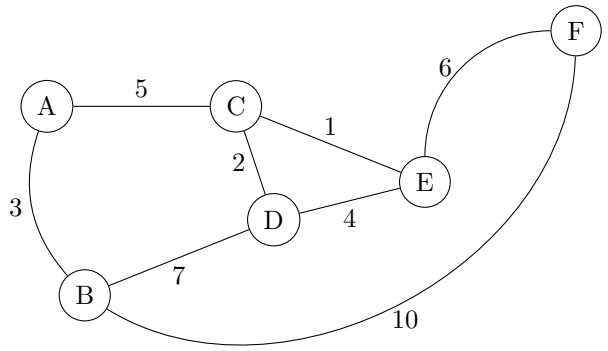
Djikstras algoritme

1. Vælg en startknude s, og sæt afstanden fra s til alle andre knuder til uendelig, ∞. Markér s som besøgt (en besøgt knude bliver aldrig besøgt igen).
2. Udregn afstanden fra s til den nuværende knudes naboer. Hvis afstanden er kortere end den tidligere noterede afstand, erstat den gamle afstand med den nye.
3. Udvælg knuden (der ikke nødvendigvis er en nabo til den nuværende knude) med kortest afstand til s og markér denne som besøgt. Denne knude bliver også den nye nuværende knude.
4. Gentag trin 2 og 3 til alle knuder er besøgt.

Bemærkning 3.1. Djikstras algoritme giver ikke selve vejen som output, kun længden af den korteste vej. Dog er det ofte let at aflæse selve vejen ud fra ens udregninger.

Det kan være noget af en mundfuld, når man støder på sådan en algoritme for første gang. Algoritmen forstås bedst gennem nogle eksempler. Vi tager et nu:

Eksempel 3.2. Betragt den vægtede graf:



Figur 10: En vægtet grad med 6 punkter.

Vi anvender Djikstras algoritme til at finde den korteste vej fra A til alle andre knuder. A er vores startknude, så den er implicit markeret som besøgt. Dette er første trin i algoritmen. Vi ser, at A er nabo til knude B og C med afstand 3 og 5. Vi skriver afstandene ned i et skema, hvor afstanden til A står under hver knude:

Nuværende knude	B	C	D	E	F
$A \rightarrow$	3	5	∞	∞	∞

B har kortest afstand til A. Vi markerer derfor B som besøgt. B har veje til D og F af længde 7 og 10. Vi kopierer disse oplysninger ind i skemaet:

Nuværende knude	B	C	D	E	F
$A \rightarrow$	3	5	∞	∞	∞
$B \rightarrow$	✓	5	$3+7$	∞	$3+10$

Husk, at tallene under knuderne er afstanden til A med den vej, vi har fundet! Det ses, at knude C har kortest afstand til A, så den fortsætter vi med. C har en vej til D og E med længde 2 og 1. Den forrige vej til D havde længde $3 + 7 = 10$, men vi har fundet en ny vej med længde $5 + 2 = 7$. Vi skriver de nye oplysninger ind i skemaet:

Nuværende knude	B	C	D	E	F
$A \rightarrow$	3	5	∞	∞	∞
$B \rightarrow$	✓	5	$3 + 7$	∞	$3 + 10$
$C \rightarrow$		✓	$5 + 2$	$5 + 1$	13

Det ses, at E er knuden med kortest vej til A, som vi endnu ikke har besøgt. E har en vej til F med længde 6, så vi har fundet en kortere vej til F end før. Vi skriver oplysningerne ind:

Nuværende knude	B	C	D	E	F
$A \rightarrow$	3	5	∞	∞	∞
$B \rightarrow$	✓	5	$3 + 7$	∞	$3 + 10$
$C \rightarrow$		✓	$5 + 2$	$5 + 1$	13
$E \rightarrow$			7	✓	$6 + 6$

Vi ser, at D er den ikke-besøgte knude, der har kortest afstand til A. D har dog ingen naboer, vi endnu ikke har besøgt. Vi markerer derfor D som besøgt og hopper videre:

Nuværende knude	B	C	D	E	F
$A \rightarrow$	3	5	∞	∞	∞
$B \rightarrow$	✓	5	$3 + 7$	∞	$3 + 10$
$C \rightarrow$		✓	$5 + 2$	$5 + 1$	13
$E \rightarrow$			7	✓	$6 + 6$
$D \rightarrow$			✓		12

Der er kun F tilbage, som vi markerer som besøgt:

Nuværende knude	B	C	D	E	F
$A \rightarrow$	3	5	∞	∞	∞
$B \rightarrow$	✓	5	$3 + 7$	∞	$3 + 10$
$C \rightarrow$		✓	$5 + 2$	$5 + 1$	13
$E \rightarrow$			7	✓	$6 + 6$
$D \rightarrow$			✓		12
$F \rightarrow$					✓

Da der ikke er flere knuder tilbage at tjekke, er vi færdige! Vi aflæser de nederste værdier i skemaet under hver knude og får, at den korteste vej fra:

- A til B er 3
- A til C er 5
- A til D er 7

- A til E er 6
- A til F er 12

I starten af sektionen hævdede vi, at Djikstras algoritme altid giver den korteste vej i en vægtet graf med positive vægte. Dette er et resultat, vi fremhæver her (dog uden bevis):

Sætning 3.3. *Lad G være en vægtet graf, hvor alle vægte er positive, med en knude s. Djikstras algoritme giver den korteste vej fra s til alle andre knuder.*

Bevis. Beviset udelades, idet det er for langt til, at vi kan gennemgå det her. Et klassisk bevis, der benytter såkaldte løkke-invarianter, kan findes i [1]. ■

3.3 Kruskals algoritme og mindste udspændende træer

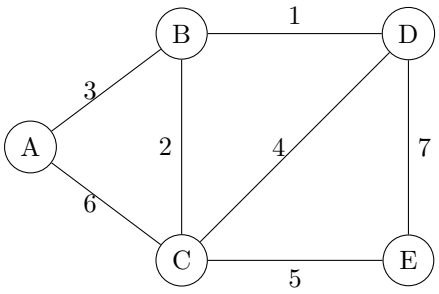
Her gennemgås Kruskals algoritme til at finde et mindste udspændende træ i en graf. Overordnet fungerer algoritmen ved at tilføje kanter efter lavest vægt én for én til et træ, der ender med at udspænde grafen. Proceduren er:

Kruskals algoritme

1. Sortér kanterne i grafen efter vægt fra mindst til størst. Har nogle af kanterne ens vægt, er rækkefølgen ligegyldig. Lad A betegne mængden (som i starten er tom) af de kanter, der skal være med i vores træ.
2. Kig på kanten med mindst vægt. Hvis tilføjelsen af kanten til A skaber en kreds i A, spring kanten over. Ellers tilføj kanten til A.
3. Gentag trin 2 for den næste kant i den sorterede liste af kanter, indtil alle kanter er blevet checket.

Vi tager naturligvis et eksempel:

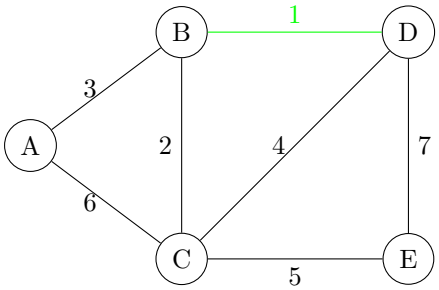
Eksempel 3.4. Betragt den vægtede graf:



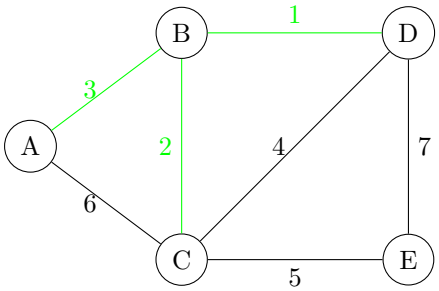
Figur 11: En vægtet graf med 5 knuder

Vi benytter Kruskals algoritme til at finde et mindste udspændende træ. Kanten, vi betragter i et givent trin, farves rød, mens kanterne, der tilføjes, farves grønne.

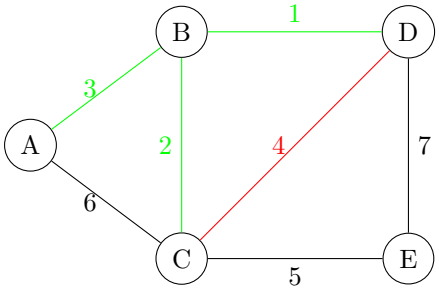
Kanten mellem B og D har mindst vægt, så den tilføjes til mængden A.



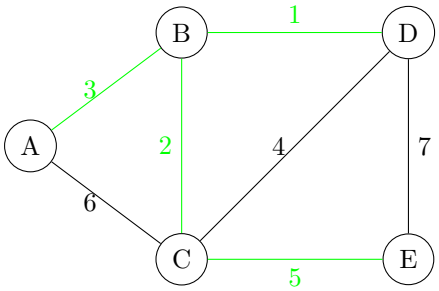
Den næste kant, vi ser på, er mellem B og C med vægt 2. Vi får ingen kreds ved at tilføje denne til A, så det gør vi. Det samme ser vi for kanten mellem A og B, så den tilføjer vi også og får:



Den næste kant er kanten mellem C og D:



Tilføjes denne kant, ser vi, at der opstår en kreds gennem B, C og D. Ergo tilføjer vi ikke denne kant. Herefter ser vi på kanten mellem C og E med vægt 5. Vi får ingen kreds ved at tilføje denne kant, så det gør vi:



Tilføjelsen af flere kanter giver en kreds, idet vi allerede har et udspændende træ. Ergo er vi færdige, og det mindste udspændende træ har en samlet vægt på $1 + 2 + 3 + 5 = 11$.

Vi konkluderer med følgende resultat:

Sætning 3.5. *Kruskals algoritme giver altid et mindste udspændende træ.*

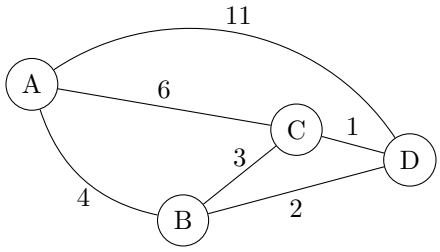
Bevis. Vi refererer til [1] for et bevis. ■

3.4 Opgaver til vægtede grafer og grafalgoritmer

Opgave 3.1 til 3.9 omhandler Djikstra's algoritme og korteste veje, mens de resterende omhandler Kruskals algoritme og mindste udspændende træer.

- **Opgave 3.1:**

1)Brug Djikstra's algoritme til at finde længden af den korteste vej fra A til D i følgende graf:



og bestem selve den korteste vej.

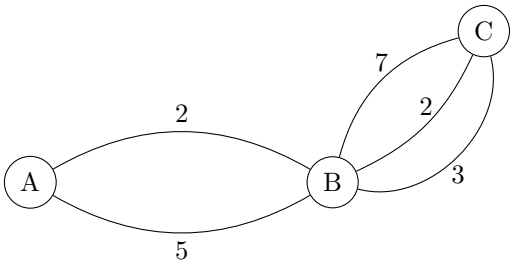
2)Bestem selve de korteste veje fra A til alle andre knuder i grafen fra eksempel 3.2.

- **Opgave 3.2:**

Giv et eksempel på en vægtet graf, hvor den korteste vej mellem to knuder ikke er entydig, altså der er flere forskellige korteste veje. Vælg din yndlingsknude og anvend Djikstra's algoritme på din graf.

- **Opgave 3.3:**

I denne opgave skal I overveje, hvad vi skal gøre mht. Djikstra's algoritme, hvis en graf har to eller flere veje mellem to knuder, f.eks.:



1)Når vi skal finde korteste veje, hvorfor kan vi så bare antage, at en vægtet graf kun har én vej mellem to naboknuder?

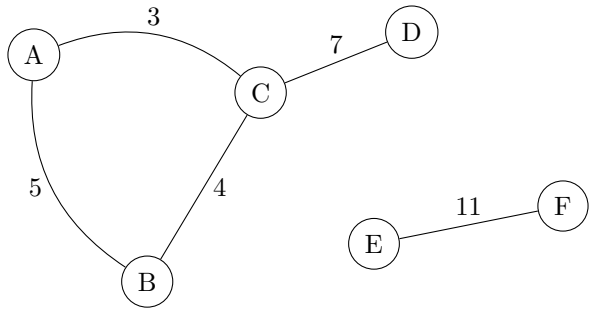
2)Hvad gør vi, hvis en graf har knuder med løkker?

3)Hvorfor kan vi uden problemer antage i dette afsnit, at alle grafer er simple?

- **Opgave 3.4:**

1)Kan vi stadig bruge Djikstra's algoritme på en ikke-sammenhængende graf? Og i så fald, hvordan?

2)Se grafen herunder:

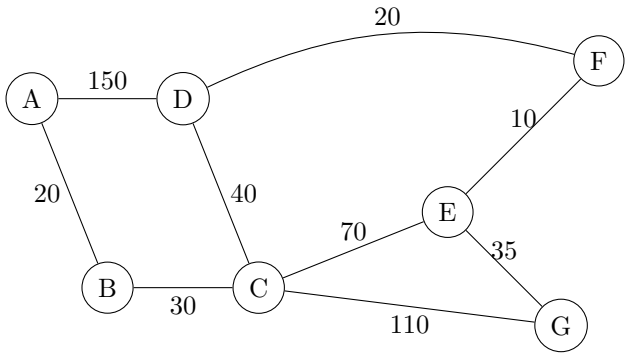


Med overvejelserne fra 1) i tankerne, hvad er den korteste afstand fra A til F?

3)Overvej til slut, hvorfor vi mon sætter afstanden fra startknuden til de andre knuder til uendelig ∞ i starten af algoritmen.

●● **Opgave 3.5:**

Superman befinder sig i by A og har hørt, at der er problemer i by G. Godt nok flyver Superman hurtigt, men han vil stadig gerne tilbagelægge så kort en afstand som muligt. Heldigvis kender Superman alle ruter mellem byerne. Ud fra følgende graf, find længden af den korteste vej fra A til G samt selve vejen:



●● **Opgave 3.6:**

Tegn en vægtet graf, hvor Djikstra's algoritme **ikke** giver den korteste vej.

●●● **Opgave 3.7:**

1)Antag, at vi arbejder med en vægtet graf, hvor alle vægte er positive eller 0. Bevis, at en korteste vej ikke kan indeholde en kreds, medmindre kredsens samlede vægt er 0 (så alle kanter i kredsen har vægt 0).

2)Tegn en sammenhængende vægtet graf (ikke nødvendigvis positive vægte), hvor en korteste vej mellem to knuder ikke eksisterer. [Vink: kan du tegne en graf, hvor man kan gøre afstanden mellem to knuder arbitrært lille?]

●●● **Opgave 3.8:**

Bevis følgende påstande for en vægtet graf med positive vægte, G, hvori A og B er

knuder (du må ikke benytte Djikstra's algoritme i dit argument!):

1)Hvis der findes en vej fra A til B, findes også en kortest mulig vej fra A til B. [Vink: brug forrige opgave!]

2)Antag, at den korteste vej fra A til B går gennem en knude C. Da er den del af vejen fra A til B, som går mellem A og C, den korteste vej fra A til C. Med andre ord: korteste veje er opbygget af korteste veje.

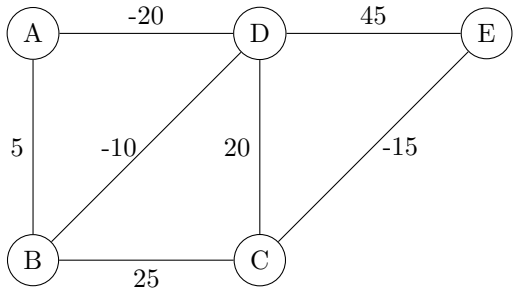
Note: Delopgave 2) viser, at problemet med at finde korteste veje i en graf har såkaldt *optimal delstruktur*. Det betyder, at en optimal løsning til problemet består af optimale løsninger til delproblemer.

•• **Opgave 3.9:**

Tegn selv en vægtet graf og find længden af den korteste vej mellem din yndlingsknude og alle andre knuder.

• **Opgave 3.10:**

1)Find et mindste udspændende træ i følgende graf:



Hvad er den minimale vægt for et udspændende træ?

2)Hvis vægten af kanten fra C til D ændres til 30, er dit udspændende træ fra opgave 1) stadig et mindste udspændende træ?

• **Opgave 3.11:**

Tegn en vægtet graf med mindst 6 knuder og 9 kanter, hvor der findes mere end ét mindste udspændende træ.

• **Opgave 3.12:**

Hvis vi dropper kravet om at sortere kanterne efter vægt i starten af Kruskals algoritme og blot betragter dem i en vilkårlig rækkefølge, får vi så stadig altid et mindste udspændende træ, når algoritmen terminerer?

•• **Opgave 3.13:**

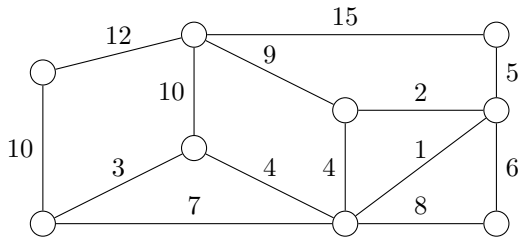
Lad A udgøre en samling af kanter i en vægtet graf G, der forbinder samtlige knuder i G og som har minimal samlet vægt.

1)Bevis, at hvis alle kanter i G har positiv vægt, er A nødvendigvis et træ.

2)Gælder det samme, hvis vi tillader kanterne i G at have negativ vægt? Enten giv et bevis eller et modeksempel.

•• Opgave 3.14:

Du er ansat på en fabrik, der fremstiller elektroniske komponenter. Du får udleveret følgende skitse af en chip:



Figur 12: En skitse af chippen

Kanterne skal her forstås som mulige forbindelser, der kan laves med kobbertråd. Vægtene er længden mellem knuderne (jo længere, jo mere kobber skal der bruges for at lave den forbindelse). Fabrikken vil have, at alle punkter på chippen skal indgå i ét netværk. Find sådan et netværk, hvor der skal bruges så lidt kobber som muligt. Hvor meget kobber skal der bruges i netværket?

••• Opgave 3.15:

Lad G være en sammenhængende vægtet graf med en kant e. Antag, at e indgår i en kreds og har maksimal vægt af alle kanterne i kredsen. Bevis, at der eksisterer et mindste udspændende træ, der ikke indeholder e.

••• Opgave 3.16:

Lad G være en sammenhængende vægtet graf med naboknuder A og B. Antag, at kanten fra A til B har minimal vægt blandt kanterne i grafen. Bevis, at denne kant vil tilhøre et mindste udspændende træ for G.

4 Litteraturliste

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. Massachusetts Institute of Technology, 3 edition, 2009. ISBN 978-0-262-53305-8.

[2] Jesper Lützen. *Diskrete Matematiske Metoder*. Københavns Universitet, 2 edition, 2019.

[3] Wikimedia. Image-koenigsberg - wikimedia, 2016. URL `https://commons.wikimedia.org/wiki/File:Image-Koenigsberg,_Map_by_Merian-Erben_1652.jpg`.