Grafdefinitioner

$N = \{i\}$: noder (hörn)

$B = \{(i,j)\}, i \in N, j \in N$: bågar (kanter)

Graf: $G = (N, B)$

Definitioner

Väg: Sekvens av angränsande bågar.

Cykel: Väg som startar och slutar i samma nod.

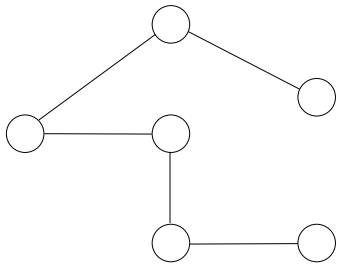
En *enkel* väg innehåller ingen cykel.

En *enkel* cykel innehåller ingen mindre cykel.

En **sammanhängande** graf har en väg mellan varje par av noder.

Träd: Sammanhängande graf utan cykler.

Träd



Träd

Sats

För ett träd med n (> 1) noder gäller:
- Det har $n - 1$ bågar.
- Mellan varje par av noder finns en unik väg.
- Om en ny båge läggs till skapas exakt en cykel.
- Om en båge tas bort bildas två träd.

Uppspännande träd

Sats

För en graf med n noder är följande begrepp ekvivalenta:
- Ett uppspännande träd.
- En sammanhängande graf med $n - 1$ bågar.
- En graf utan cykler med $n - 1$ bågar.

En graf är sammanhängande om och endast om den innehåller ett uppspännande träd.

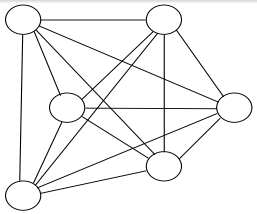
Definition

Skog: Graf utan cykler (dvs ett eller flera träd).

Grafer

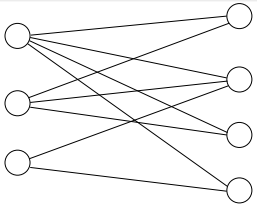
Definition

En **fullständig graf** har en båge mellan varje par av noder.



Definition

I en **tudelad graf** går alla bågar från en nodmängd till en annan.

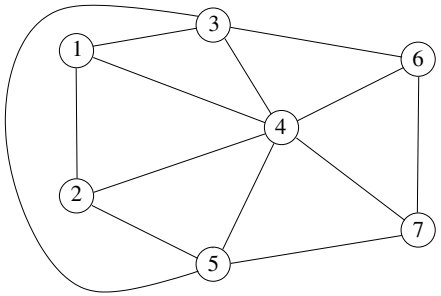


Grafer

Definition

En **plan** graf kan ritas så att inga bågar korsar varandra (förutom i noderna).

Plan? Ja, plan.



Cykler och vägar

Definitioner

En **Eulercykel** är en cykel som använder varje *båge* exakt en gång.

En **Hamiltoncykel** är en cykel som passerar varje *nod* exakt en gång.

En **Hamiltonväg** är en väg som passerar varje *nod* exakt en gång.

Exempel på frågeställningar:

Finns en Eulercykel i en given graf?

Finns en Hamiltoncykel i en given graf?

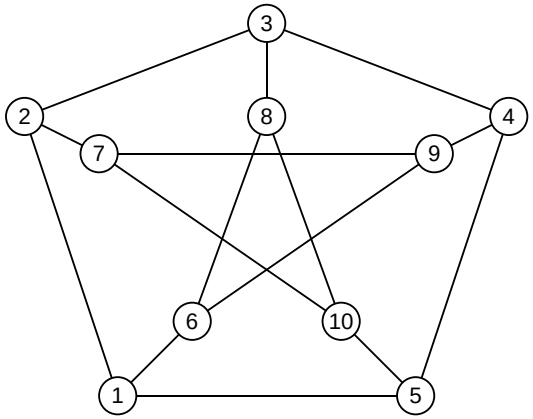
Definition

En nods **valens** är antalet bågar som ansluter till noden.

Sats

En sammanhängande oriktad graf har en Eulercykel om och endast om alla dess noder har jämn valens.

Hamiltoncykel



Petersens graf saknar Hamiltoncykel och Eulercykel.

Notation: mängder

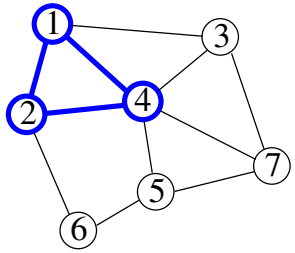
$\delta(i)$: bågarna som ansluter till nod i.
$\delta^+(i)$: bågarna som kommer in till nod i
$\delta^-(i)$: bågarna som går ut ur nod i.

$\delta^-(A)$: de bågar som går ut ur A.
$\delta^+(A)$: de bågar som går in till A
$\delta(A)$: de bågar som har en av sina ändnoder i A.

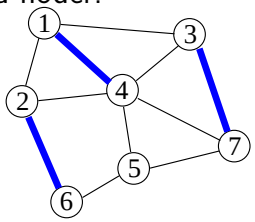
$\gamma(A)$: de bågar som har båda sina ändnoder i A.

Nodmängder och Bågmängder

En **klick** är en fullständig subgraf.

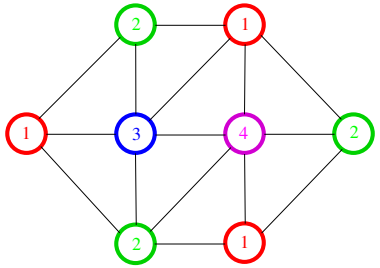


Matchning: Varje nod ansluter till högst en båge. En *perfekt* matchning innehåller alla noder.

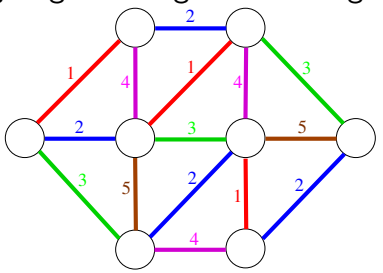


Graffärgning

Nodfärgning: Inga två närliggande noder har samma färg.



Bågfärgning: Inga två angränsande bågar har samma färg.



Intressanta optimeringsproblem

- Maximal klick, n_K.
- Maximal matchning, m_M.
- Nodfärgning med min antal färger, χ.
- Bågfärgning med min antal färger, χ'.

Satser

För tudelad graf: $\chi = 2$.

$\chi \geq n_K$. (En graf kallas *perfekt* om $\chi = n_K$.)

Noderna i varje plan graf kan färgas med fyra färger.

$v_{MAX} \leq \chi' \leq v_{MAX} + 1$. ($v_{MAX}$ är grafens maxvalens.)

$\chi' = 1$ om och endast om grafen är en matchning.

$\chi' = v_{MAX}$ om grafen är tudelad.

$\chi' = v_{MAX}$ om grafen är plan, saknar parallella bågar och har $v_{MAX} \geq 7$.

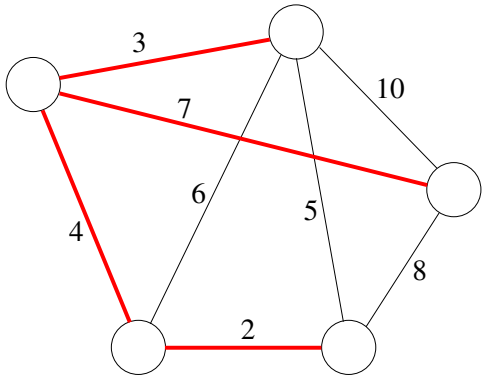
χ' är v_{MAX} eller $v_{MAX} + 1$, men det är *NP*-fullständigt att avgöra vilket.

Metoder? Heuristiker. (Kommer senare.)

Billigaste uppspännande träd (MST)

Koppla ihop några datorer på billigaste sätt.

Finn ett billigaste uppspännande träd i en given oriktad graf med bågkostnader.



Heltalsformulering av MST

Variabeldefinition:
$x_{ij} = 1$ om båge (i,j) ingår i trädet,
0 om inte, för alla $(i,j) \in B$.

$$\begin{aligned}
\text{min} \quad & \sum_{(i,j)\in B} c_{ij}x_{ij} \\
\text{då} \quad & \sum_{(i,j)\in B} x_{ij} = |N| - 1 \\
& \sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \quad \text{för alla } S \subset N \quad \text{(Inga cykler.)} \\
& x_{ij} \in \{0,1\} \quad \text{för alla } (i,j) \in B
\end{aligned}$$

Alternativt: $\sum_{i\in S}\sum_{j\in N\setminus S} x_{ij} \geq 1$ för alla $S \subset N$ (Sammanhängade.)

Många bivillkor. (Exponentiellt många.)

Graf-baserade lösningsmetoder:

Ett uppspännande träd har $n-1$ bågar, samt

- saknar cykler
- är sammanhängande

Två metodprinciper:

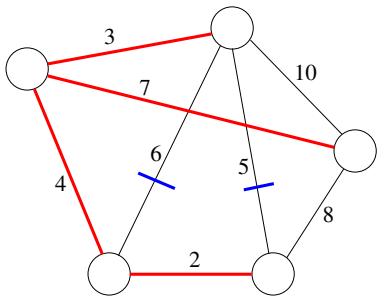
1. Ta med billigaste återstående bågen, om ingen cykel bildas (Kruskals metod).

2. Ta med billigaste bågen som utökar delträdet (Prims metod).

Kruskals metod:

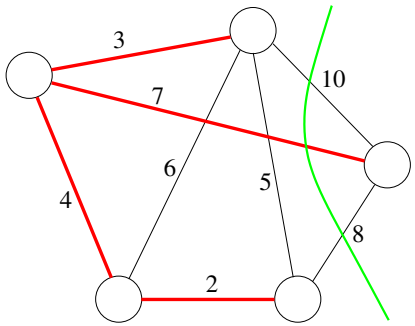
1. Finn billigaste återstående båge. Ta bort bågen ur listan.
2. Om bågen ej bildar cykel, ta med den. (Annars släng den.)
3. Om antal bågar är $n-1$: Stopp. Annars gå till 1.

Komplexitet: $O(|B| \log |B|)$ (sortera bågarna först)



Prims metod:

0. Låt nod 1 vara delträdet.
1. Finn billigaste båge ut från delträdet.
2. Ta med den och dess andra ändnod i delträdet.
3. Om antal bågar är $n-1$: Stopp. Annars gå till 1.



Prims metod (bättre implementering)

Håll reda på trädets närmaste granne till nod i, w_i.

Spara möjliga nodmärkningar.

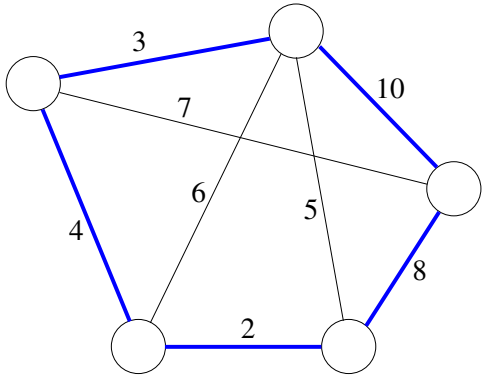
0. Låt nod 1 vara delträdet och sätt $w_i = 1$ för alla andra noder i.
1. Finn billigaste båge ut ur delträdet: $\min_i c_{i,w_i}$.
3. Ta med bågen i delträdet.
4. Uppdatera w_i via den nya bågen. (Närmare granne?)
5. Om antal bågar är $n-1$: Stopp. Annars gå till 1.

Komplexitet: $O(|N|^2)$

Handelsresandeproblemet (TSP)

Finn kortaste rundtur som besöker alla platser/noder.

Finn en billigaste Hamiltoncykel i en given graf med bågkostnader.



Handelsresandeproblemet med återbesök (TSPr):
Finn billigaste cykel i grafen som besöker varje nod *minst* en gång.

Handelsresandeproblemet: varianter

Specialfall: ΔTSP: Symmetrisk kostnadsmatris som uppfyller triangelolikheten:
$c_{ij} \leq c_{ik} + c_{kj}$ för alla k, i, j.
(Implicerar fullständig graf.)

Specialfall: ΔTSPb: Symmetrisk kostnadsmatris som uppfyller den begränsade triangelolikheten:
$c_{ij} \leq c_{ik} + c_{kj}$ för alla bågar som finns.

ΔTSP har samma lösning som ΔTSPr. (Återbesök lönar sig aldrig.)

Heltalsformulering av TSP, riktad graf

Variabeldefinition:
$x_{ij} = 1$ om båge (i,j) ingår i cykeln, 0 om inte.

$$\min \sum_{(i,j)\in B} c_{ij}x_{ij}$$

$$\text{då} \quad \sum_{i\in N} x_{ij} = 1 \quad \text{för alla } j \quad \text{(en in till varje nod)}$$

$$\sum_{j\in N} x_{ij} = 1 \quad \text{för alla } i \quad \text{(en ut från varje nod)}$$

$$\sum_{i\in S}\sum_{j\in N\setminus S} x_{ij} \geq 1 \quad \text{för alla } S \subset N \quad \text{(sammanhängande)}$$

$$x_{ij} \in \{0,1\} \quad \text{för alla } (i,j) \in B$$

eller

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \quad \text{för alla } S \subset N \quad \text{(inga småcykler)} \quad \text{(ej TSPr)}$$

TSP, oriktad graf

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in B} c_{ij}x_{ij} & & & \\
& \sum_{i\in N} x_{ij} & = & 2 & \text{för alla } j \quad \text{(valens två)} \\
& \sum_{i\in S}\sum_{j\in N\setminus S} x_{ij} & \geq & 2 & \text{för alla } S \subset N \quad \text{(sammanhängande)} \\
& x_{ij} \in \{0,1\} & & & \text{för alla } (i,j) \in B
\end{aligned}$$

För TSPr: Ta bort första bivillkoren.

Alternativ (ej för TSPr):

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \quad \text{för alla } S \subset N \quad \text{(inga småcykler)}$$

Observera likheten i formulering med MST. TSP är dock *mycket* svårare.

Relaxation av TSP: 1-träd

Billigaste 1-träd: Ett MST för noderna $\{2, 3, \ldots, n\}$, plus de två billigaste bågarna som ansluter till nod 1.

(Alltså inte ett träd.)

Rätt antal bågar.

Nod 1 får valens 2, men andra noder kan ha fel valens.

En cykel (som kan vara för liten) bildas genom nod 1.

Lika lätt att hitta billigaste 1-träd som MST.

Relaxation: Ger undre gräns/optimistisk uppskattning.

Ett 1-träd är en Hamiltoncykel om varje nods valens är lika med två.

Om billigaste 1-träd är en Hamiltoncykel, är turen optimal.

Förbättring av 1-träd

Nodstraff:

Addition av konstant till c för alla bågar som ansluter till en nod ändrar ej optimal handelsresandetur.

Alla tillåtna lösningar blir $2c$ dyrare.

1. Finn billigaste 1-träd.
2. Stopp om Hamiltoncykel fås. Optimum.
3. Välj en nod med för hög valens. Öka kostnaderna för alla bågar som ansluter till noden med t.ex. 1.
4. Ge upp eller gå till 1.

Försök få tillåten tur

Utgå från en lösning där alla noder inte har valens två, t.ex. ett billigaste 1-träd.

Mål: Försök få en tillåten lösning genom att byta enstaka bågar.

Byt en ändnod för enstaka bågar, från noder med för hög valens till noder med för låg.

För att behålla en sammanhängande lösning: Gör förändring i cykler.

Lägg till en båge till en nod med för låg valens.

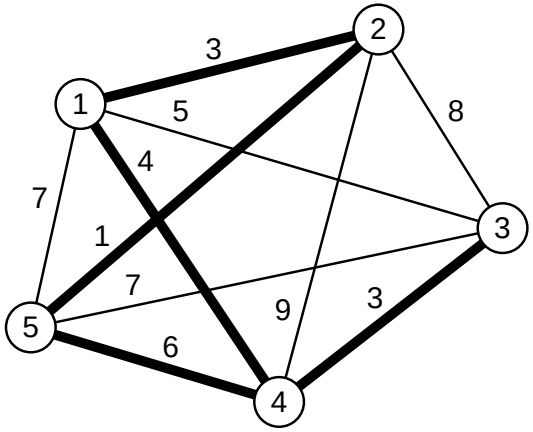
Då bildas alltid en cykel.

Ta bort en båge i cykeln, så att valenserna förbättras (och kostnaden inte ökar för mycket).

Upprepa några gånger.

Inget garanterat resultat.

Exempel: 1-träd



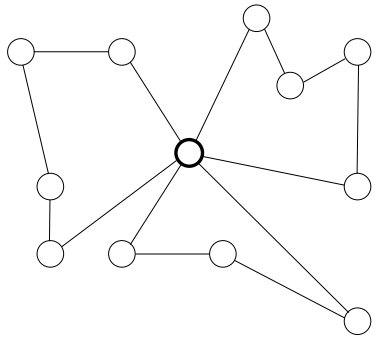
Billigaste 1-träd. Kostnad: 17. Undre gräns: 17.

Exempel: TSP: Nodstraff

1-träd. Nod med hög valens. Öka kostnader. 1-träd. Optimum. Kostnad: 18

Ruttplaneringsproblemet

Ett antal lastbilar skall köra ut varor till ett antal kunder. Varje lastbil startar i en depå, kör runt till några kunder och levererar varor och återvänder till depån. Lastbilarna har begränsad lastkapacitet och kunderna given efterfrågan.



Varje lastbil kör en handelsresandetur i en mängd noder som ska bestämmas. Samtliga noder skall täckas av någon tur.
Målfunktion: Minimera kostnaden eller avgasutsläppen eller en kombination av dem.

Matchningsproblemet

Exempel: Bilda en graf där noder är personer och en båge mellan två noder visar att personerna kan samarbeta.

Bilda så många par som möjligt.

Matchning: Högst en av bågarna ansluter till någon nod.

Matchning med maximal kardinalitet (max antal bågar):

$$\begin{aligned} \max \quad & \sum_i \sum_j x_{ij} \\ \text{då} \quad & \sum_j (x_{ij} + x_{ji}) \leq 1 \quad \text{för alla } i \\ & x_{ij} \in \{0,1\} \quad \text{för alla } i,j \end{aligned}$$

Matchning med maximal vikt:

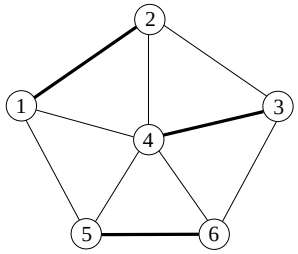
$\max \sum_i \sum_j c_{ij} x_{ij}$ under samma bivillkor

Matchningsproblemet

Perfekt matchning: samtliga noder har valens ett.

Perfekt matchning med minimal vikt:

$$\begin{aligned} \min \quad & \sum_i \sum_j c_{ij} x_{ij} \\ \text{då} \quad & \sum_j (x_{ij} + x_{ji}) = 1 \quad \text{för alla } i \\ & x_{ij} \in \{0,1\} \quad \text{för alla } i,j \end{aligned}$$



Metod för matchning med max kardinalitet

Alternerande väg: Varannan båge ingår i matchningen.

Utökande väg: Första och sista bågen ingår ej i matchningen.

En bättre matchning (en båge mer): Byt matchad båge mot omatchad längs en utökande väg.

Sats (Berge 1957)

En matchning är maximal om och endast om det inte finns någon utökande väg.

Metod: Så länge det finns minst två omatchade noder:

Från varje omatchad nod: Finn en utökande väg. Byt matchning längs vägen.

Antingen finner man en utökande väg, och noden blir matchad, eller så finner man inte det, vilket bevisar att noden inte kan matchas.

Metod för matchning med max kardinalitet

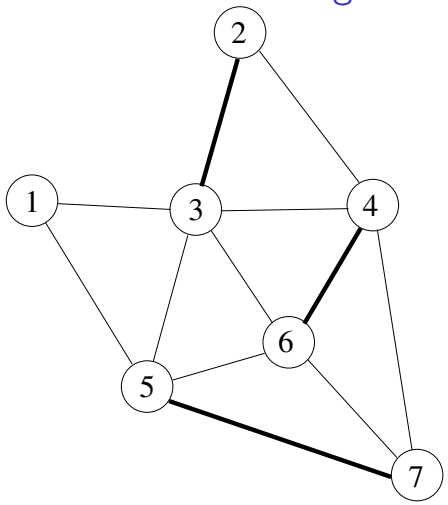
Hur finner man utökande väg?

1. Välj en omatchad nod, i_1, och märk den med $-$.
2. Välj en omatchad båge (i,j), där nod i är märkt med $-$ och nod j omärkt. Märk nod j med $+$.
3. Välj en matchad båge (i,j), där nod i är märkt med $+$ och nod j omärkt. Märk nod j med $-$.
4. Upprepa 2 och 3 tills en omatchad nod, j_1, blir märkt med $+$.
5. Nysta upp en alternerande väg från i_1 till j_1.

Obs: Vägen kan antingen vara en enda omatchad båge, eller måste innehålla minst en matchad båge.

Om det inte finns någon båge i steg 4, måste man upprepa steg 3.

Metod för matchning med max kardinalitet



En matchning. Välj en omatchad nod. Omatchad båge (7,5). Matchad båge (5,6). Omatchad båge (6,4). Matchad båge (4,3). Omatchad båge (3,2). Utökande väg: 7 - 5 - 6 - 4 - 3 - 2. Byt. En bättre matchning. Maximal, ty bara en omatchad nod.

Metod för matchning med max kardinalitet

Att leta efter en utökande väg:

Lätt för tudelade grafer. (Jämför tillordningsproblemet.)

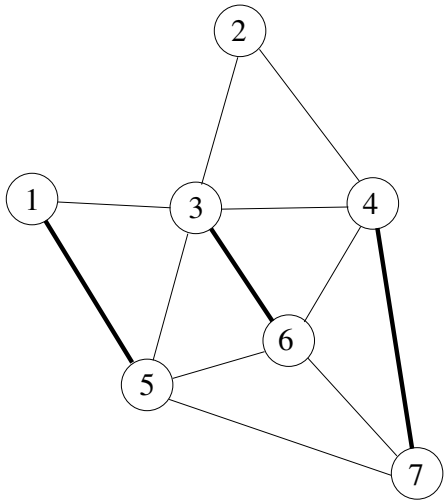
Men om inte tudelat: Kan innehålla udda cykel.

En udda cykel med k bågar och $(k-1)/2$ matchade bågar kallas "blomma".
Kan inte blir bättre.

Edmonds metod: Krymp ihop blomman till en nod.
Fortsätt som förut.
När utökande väg finnes: Expandera blomman. Ev. behöver matchningen i den vridas.

Polynomisk optimerande metod.

Metod för matchning med max kardinalitet



En matchning. En blomma. Krymp den. Utökande väg: 1 - 5 - 7 - 346. Byt. Expandera blomman. Vrid blommans matchning. En bättre matchning.

Metod för bågfärgning

I en bågfärgning utgör bågarna som färgats med en färg en matchning.

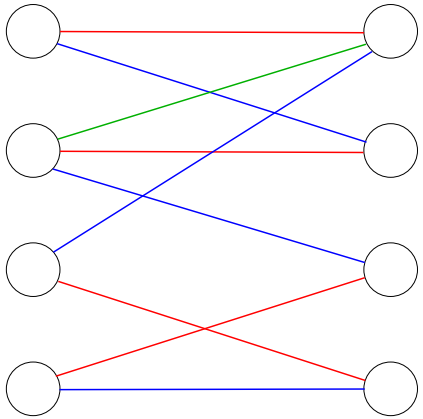
Metod:
1. Finn en matchning med maximal kardinalitet.
2. Färga matchade bågar med en färg.
3. Ta bort färgen och de matchade bågarna.

Upprepa tills inga bågar är kvar.

Matchningen behöver inte vara maximal.
Det räcker om den täcker alla noder med maximal valens.
Då minskar den maximala valensen i varje steg, så antalet iterationer är inte mer än maxvalensen (vilket är mindre än antalet noder).

Denna metod löser problemet i tudelade grafer exakt, och är en bra heuristik för andra grafer.

Metod för tudelad bågfärgning



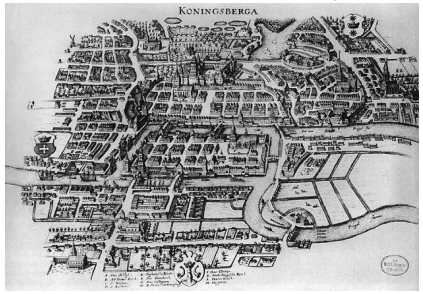
En matchning. Färga den röd. Ta bort bågarna. En ny matchning. Färga den blå. Ta bort bågarna. En ny matchning. Färga den grön. Färdig.

Kinesiska brevbärarproblemet

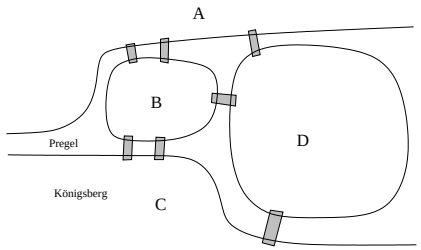
En *brevbärartur* är en tur som använder varje båge minst en gång.

Det kinesiska brevbärarproblemet är att finna en brevbärartur med minimal kostnad.

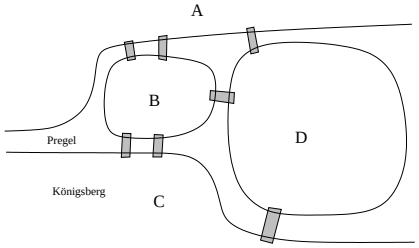
Königsbergs broar: (Euler, 1700-talet)
Försök finna en tur som passerar varje bro en gång.



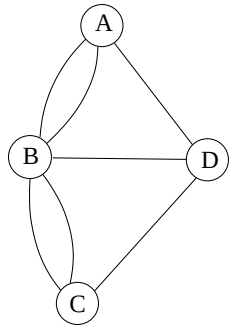
→



Kinesiska brevbärarproblemet



→



En **Eulercykel** är en cykel som använder varje *båge* i grafen exakt en gång.

En Eulercykel är en optimal brevbärartur, om den finns.

En oriktad graf har en Eulercykel om och endast om den är sammanhängande och alla dess noder har jämn valens.

Det fanns inget promenadstråk genom Königsberg som passerade varje bro exakt en gång. (Se nodvalens.)

Kinesiska brevbärarproblemet: Modell

Matematisk modell:
x_{ij}: antalet gånger båge (i,j) trafikeras.
z_i: antal gånger nod i passeras i turen.

$$\begin{aligned}
\text{min} \quad & \sum_i \sum_j c_{ij} x_{ij} & \\
\text{då} \quad & \sum_j (x_{ij} + x_{ji}) - 2z_i = 0 & \text{för alla } i \\
& x_{ij} \geq 1, \text{ heltal} & \text{för alla } i,j \\
& z_i \geq 1, \text{ heltal} & \text{för alla } i
\end{aligned}$$

Kinesiska brevbärarproblemet

Man kan visa att ingen båge behöver användas mer än en gång för mycket.

Metod:

Minimera extraarbetet.

Dvs. minimera kostnaden för de bågar som används mer än en gång.

Extraarbetet är enbart "tomkörning", så det är egentligen kostnaden för den som minimeras. Gör inget om den går fortare.

Lägg till bågar mellan noder med udda valens, så att alla noder får jämn valens, på billigaste sätt.

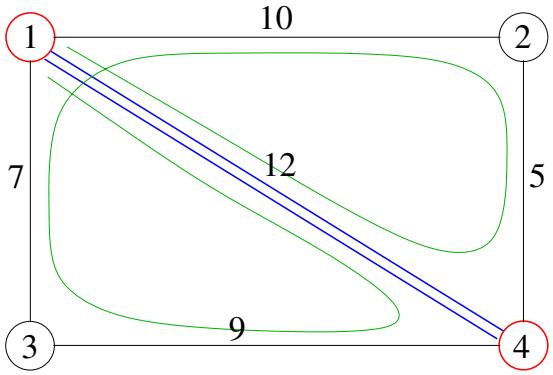
En Eulercykel ger då den billigaste brevbärarturen.

1. Förbind noderna med udda valens på billigaste sätt (mha billigaste-väg och minimal perfekt matchning).

2. Finn Eulercykeln.

Polynomisk optimerande metod.

Kinesiska brevbärarproblemet: Exempel



Finn Eulercykel. Kostnad: 43 + 12 = 55.

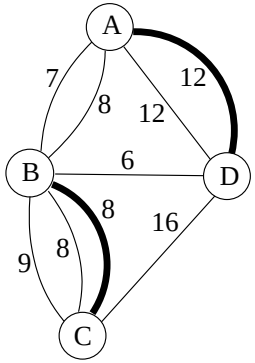
Kinesiska brevbärarproblemet

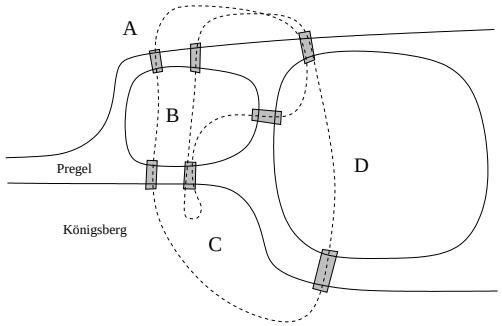
Königsberg:

Billigaste vägarna. Billigaste perfekta matchning.

Kinesiska brevbärarproblemet

Addera bågarna i billigaste perfekta matchning. Finn valfri Eulercykel.





Exempel på tur:
A-B-C-D-A-B-C-B-D-A.

En söndagspromenad i Königsberg.