

Grafdefinitioner

$N = \{i\}$: noder (hörn)

Grafdefinitioner

$N = \{i\}$: noder (hörn)

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

Grafdefinitioner

$N = \{i\}$: noder (hörn)

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

Graf: $G = (N, B)$

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.

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.

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.

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

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.

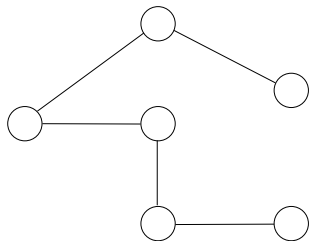
Cykel: Väg som startar och slutar i samma nod.

En *enkel* väg innehåller ingen cykel.

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

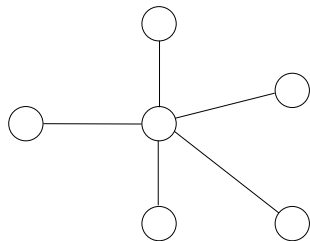
Träd: Sammanhängande graf utan cykler.

Träd



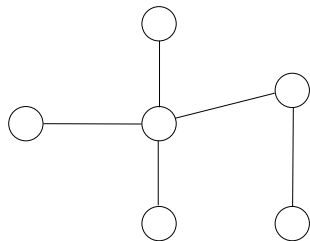
Träd

Träd



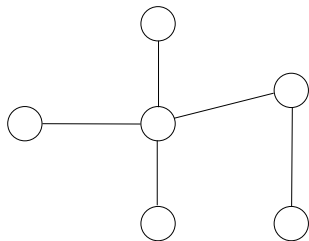
Träd

Träd



Träd

Träd



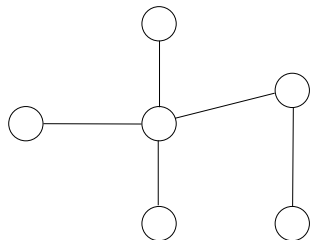
Träd

Sats

För ett träd med $n (> 1)$ noder gäller:

-
-
-
-

Träd



Träd

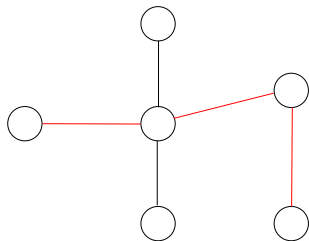
Sats

För ett träd med n (> 1) noder gäller:

- Det har $n - 1$ bågar.



Träd



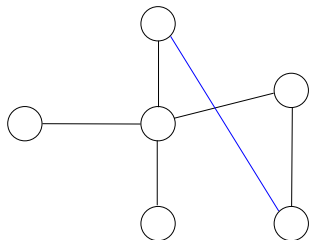
Träd

Sats

För ett träd med $n (> 1)$ noder gäller:

- Det har $n - 1$ bågar.
- Mellan varje par av noder finns en unik väg.
-
-

Träd



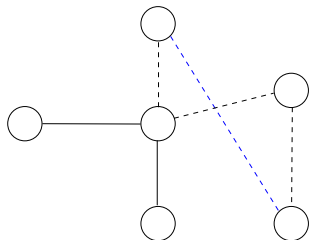
Träd

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.
-

Träd



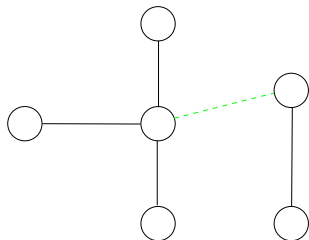
Träd

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.
-

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

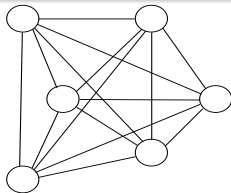
För en graf med n noder är följande begrepp ekvivalenta:

- Ett uppspännande träd.
- En sammanhängande graf med $n - 1$ bågar.
- En graf utan cykler med $n - 1$ bågar.

Grafer

Definition

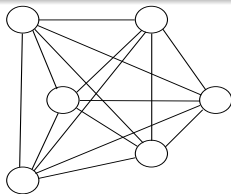
En **fullständig graf** har en båge mellan varje par av noder.



Grafer

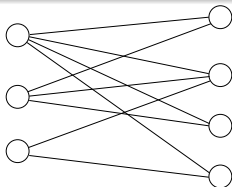
Definition

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.

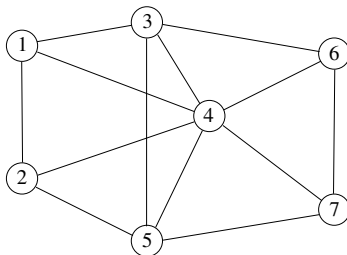


Grafer

Definition

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

Plan?

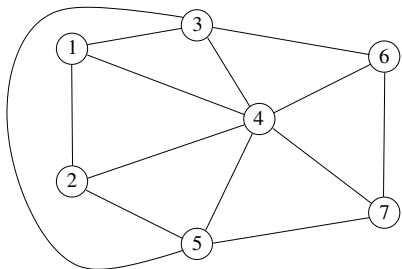


Grafer

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.

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.

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.

Exempel på frågeställningar:

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.

Exempel på frågeställningar:

Finns en Eulercykel i en given graf?

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.

Exempel på frågeställningar:

Finns en Eulercykel i en given graf?

Finns en Hamiltoncykel i en given graf?

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.

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.

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.

Exempel på frågeställningar:

Finns en Eulercykel i en given graf?

Finns en Hamiltoncykel i en given graf?

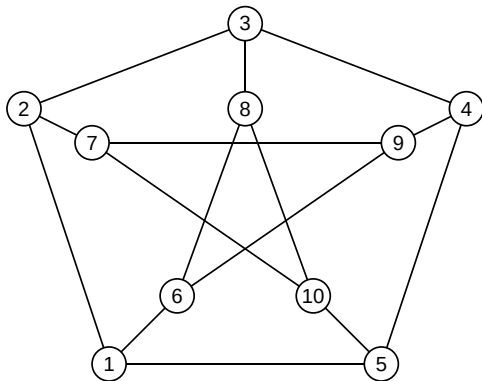
Definition

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

Sats

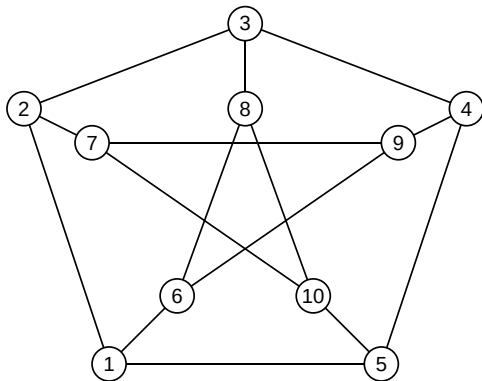
En sammanhängande oriktad graf har en Eulercykel om och endast om alla dess noder har jämn valens.

Hamiltoncykel



Petersens graf saknar Hamiltoncykel

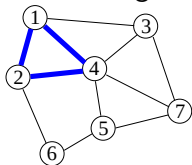
Hamiltoncykel



Petersens graf saknar Hamiltoncykel och Eulercykel.

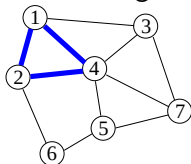
Nod-/bågmängder

En **klick** är en fullständig subgraf.

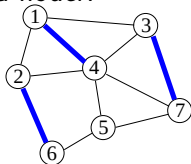


Nod-/bågmängder

En **klick** är en fullständig subgraf.

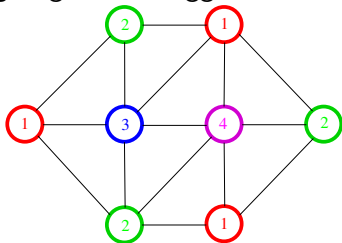


Matchning: Varje nod ansluter till högst en båge. En *perfekt* matchning innehåller alla noder.



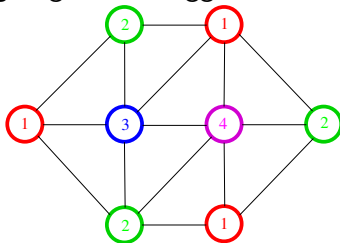
Graffärgning

Nodfärgning: Inga två närliggande noder har samma färg.

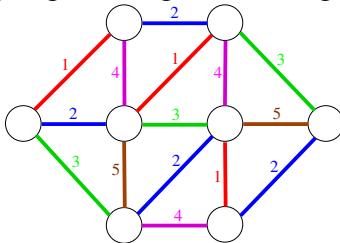


Graffärgning

Nodfärgning: Inga två närliggande noder har samma färg.



Bågfärgning: Inga två angränsande bågar har samma färg.



Intressanta optimeringsproblem

Intressanta optimeringsproblem

- Maximal klick, n_K .

Intressanta optimeringsproblem

- Maximal klick, n_K .
- Maximal matchning, m_M .

Intressanta optimeringsproblem

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

Intressanta optimeringsproblem

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

Intressanta optimeringsproblem

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

Satser

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

Intressanta optimeringsproblem

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

Satser

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

Varje plan graf kan färgas med fyra färger.

Intressanta optimeringsproblem

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

Satser

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

Varje plan graf kan färgas med fyra färger.

Satser

$\chi' \geq v_{MAX}$. $\chi' \leq v_{MAX} + 1$.

(v_{MAX} är grafens maxvalens.)

Intressanta optimeringsproblem

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

Satser

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

Varje plan graf kan färgas med fyra färger.

Satser

$\chi' \geq v_{MAX}$. $\chi' \leq v_{MAX} + 1$.

(v_{MAX} är grafens maxvalens.)

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

Intressanta optimeringsproblem

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

Satser

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

Varje plan graf kan färgas med fyra färger.

Satser

$\chi' \geq v_{MAX}$. $\chi' \leq v_{MAX} + 1$.

(v_{MAX} är grafens maxvalens.)

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

Metoder?

Intressanta optimeringsproblem

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

Satser

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

Varje plan graf kan färgas med fyra färger.

Satser

$\chi' \geq v_{MAX}$. $\chi' \leq v_{MAX} + 1$.

(v_{MAX} är grafens maxvalens.)

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

Metoder? Heuristiker.

Intressanta optimeringsproblem

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

Satser

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

Varje plan graf kan färgas med fyra färger.

Satser

$\chi' \geq v_{MAX}$. $\chi' \leq v_{MAX} + 1$.

(v_{MAX} är grafens maxvalens.)

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.

Billigaste uppspännande träd (MST)

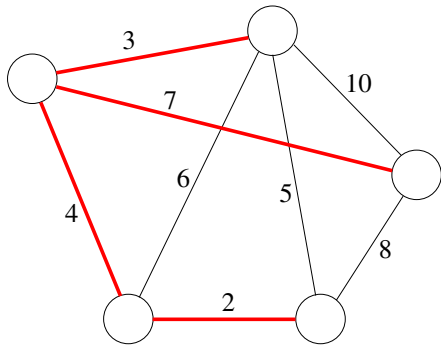
Koppla ihop några datorer på billigaste sätt.

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

Billigaste uppspännande träd (MST)

Koppla ihop några datorer på billigaste sätt.

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



Graf-baserade lösningsmetoder:

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

Graf-baserade lösningsmetoder:

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

- saknar cykler

Graf-baserade lösningsmetoder:

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

- saknar cykler
- är sammanhängande

Graf-baserade lösningsmetoder:

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

- saknar cykler
- är sammanhängande

Två metodprinciper:

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).

Graf-baserade lösningsmetoder:

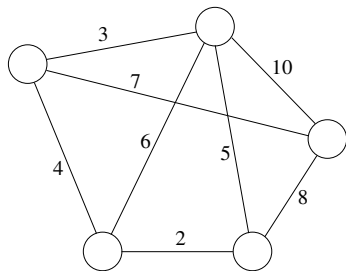
Ett uppspännande träd har $n - 1$ bågar, samt

- saknar cykler
- är sammanhängande

Två metodprinciper:

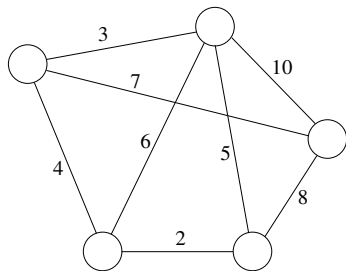
1. Ta med billigaste återstående bågen, om ingen cykel bildas (Kruskals metod).
2. Ta med billigaste bågen som utökar delträdet (Prims metod).

Kruskals metod:



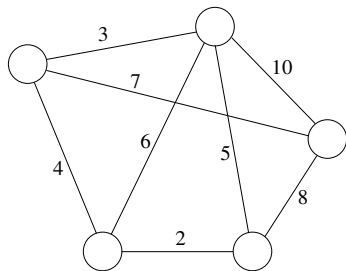
Kruskals metod:

1. Finn billigaste återstående båge. Ta bort bågen ur listan.



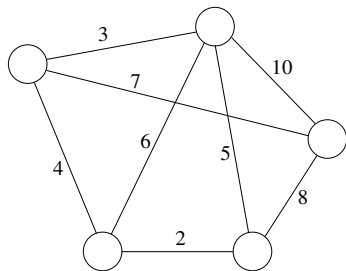
Kruskals metod:

1. Finn billigaste återstående båge. Ta bort bågen ur listan.
2. Om bågen ej bildar cykel, ta med den. (Annars släng den.)



Kruskals metod:

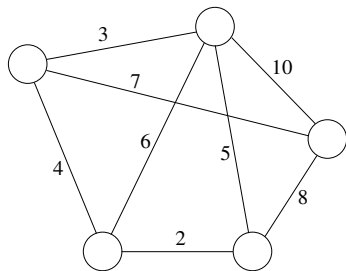
1. Finn billigaste återstående båge. Ta bort bågen ur listan.
2. Om bågen ej bildar cykel, ta med den. (Annars släng den.)
3. Om antal bågar är $n - 1$: Stopp. Annars gå till 1.



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.

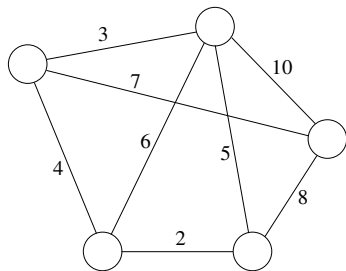
Komplexitet: $O(|B| \log |B|)$



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.

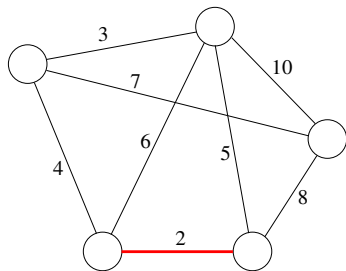
Komplexitet: $O(|B| \log |B|)$ (sortera bågarna först)



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.

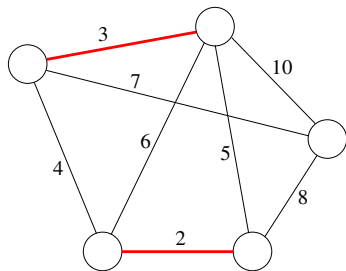
Komplexitet: $O(|B| \log |B|)$ (sortera bågarna först)



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.

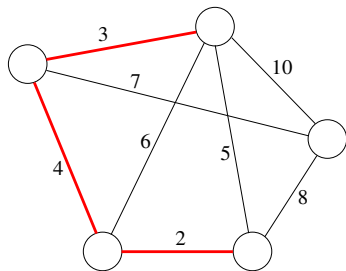
Komplexitet: $O(|B| \log |B|)$ (sortera bågarna först)



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.

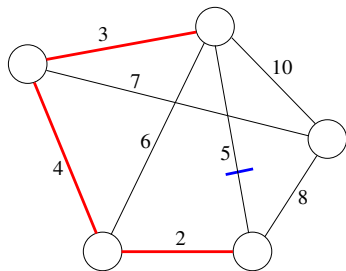
Komplexitet: $O(|B| \log |B|)$ (sortera bågarna först)



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.

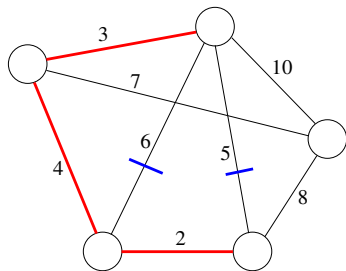
Komplexitet: $O(|B| \log |B|)$ (sortera bågarna först)



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.

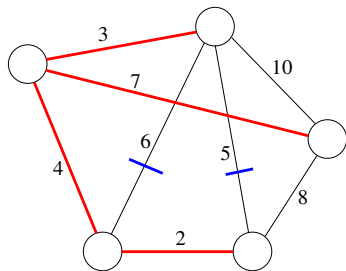
Komplexitet: $O(|B| \log |B|)$ (sortera bågarna först)



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.

Prims metod:

0. Låt nod 1 vara delträdet.
1. Finn billigaste båge ut från delträdet.

Prims metod:

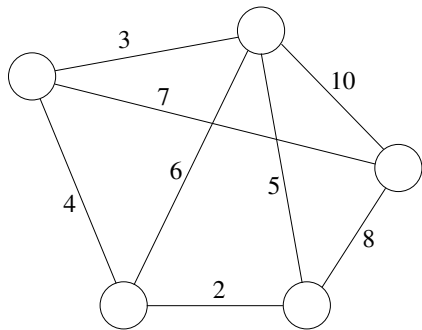
0. Låt nod 1 vara delträdet.
1. Finn billigaste båge ut från delträdet.
2. Ta med den och dess andra ändnod i delträdet.

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.

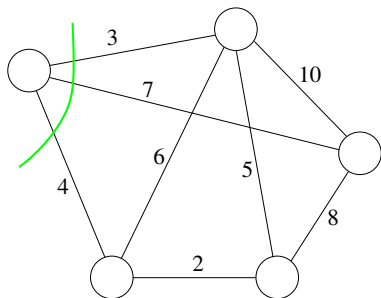
Prims metod:

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.



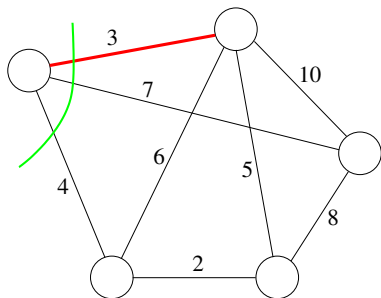
Prims metod:

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.



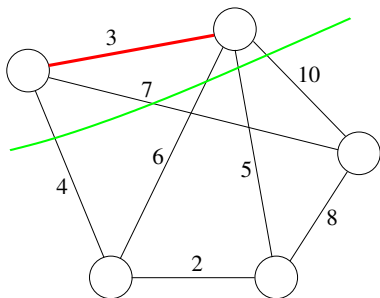
Prims metod:

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.



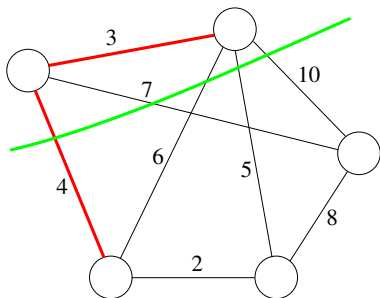
Prims metod:

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.



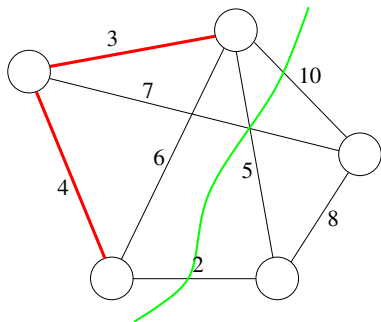
Prims metod:

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.



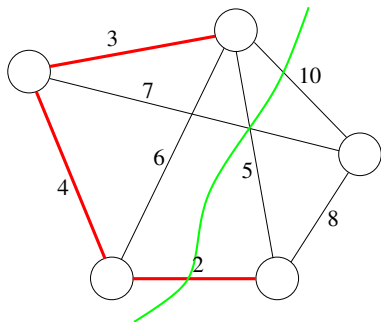
Prims metod:

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.



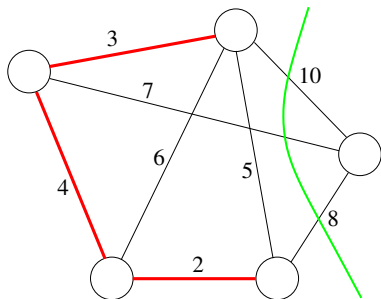
Prims metod:

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.



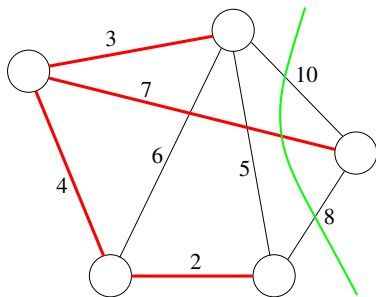
Prims metod:

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:

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 .

Prims metod (bättre implementering)

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

Spara möjliga nodmärkningar.

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 .

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: $\min_i c_{i,w_i}$.

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.

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?)

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.

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.

Handelsresandeproblemet (TSP)

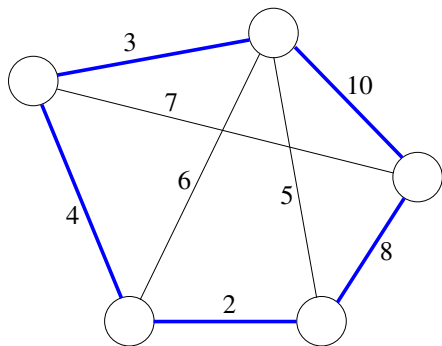
Finn kortaste rundtur som besöker alla platser/noder.

Finn en billigaste Hamiltoncykel i en given graf med bågkostnader.

Handelsresandeproblemet (TSP)

Finn kortaste rundtur som besöker alla platser/noder.

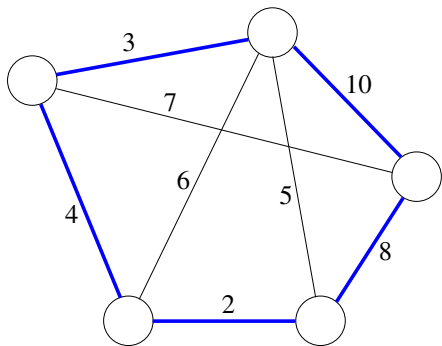
Finn en billigaste Hamiltoncykel i en given graf med bågkostnader.



Handelsresandeproblemet (TSP)

Finn kortaste rundtur som besöker alla platser/noder.

Finn en billigaste Hamiltoncykel i en given graf med bågkostnader.



Handelsresandeproblemet med återbesök (TSP_r):

Finn billigaste cykel i grafen som besöker varje nod *minst* en gång.

Handelsresandeproblemet: varianter

Specialfall: Δ TSP: Symmetrisk kostnadsmatris som uppfyller triangelolikheten:

$$c_{ij} \leq c_{ik} + c_{kj} \text{ för alla } k, i, j.$$

Handelsresandeproblemet: varianter

Specialfall: Δ TSP: Symmetrisk kostnadsmatris som uppfyller triangelolikheten:

$$c_{ij} \leq c_{ik} + c_{kj} \text{ för alla } k, i, j.$$

(Implicerar fullständig graf.)

Handelsresandeproblemet: varianter

Specialfall: Δ TSP: Symmetrisk kostnadsmatris som uppfyller triangelolikheten:

$$c_{ij} \leq c_{ik} + c_{kj} \text{ för alla } k, i, j.$$

(Implicerar fullständig graf.)

Specialfall: Δ TSPb: Symmetrisk kostnadsmatris som uppfyller den begränsade triangelolikheten:

$$c_{ij} \leq c_{ik} + c_{kj} \text{ för alla bågar som finns.}$$

Handelsresandeproblemet: varianter

Specialfall: Δ TSP: Symmetrisk kostnadsmatrix som uppfyller triangelolikheten:

$$c_{ij} \leq c_{ik} + c_{kj} \text{ för alla } k, i, j.$$

(Implicerar fullständig graf.)

Specialfall: Δ TSPb: Symmetrisk kostnadsmatrix som uppfyller den begränsade triangelolikheten:

$$c_{ij} \leq c_{ik} + c_{kj} \text{ för alla bågar som finns.}$$

Δ TSP har samma lösning som Δ TSPr. (Återbesök lönar sig aldrig.)

Handelsresandeproblemet: varianter

Specialfall: Δ TSP: Symmetrisk kostnadsmatrix som uppfyller triangelolikheten:

$$c_{ij} \leq c_{ik} + c_{kj} \text{ för alla } k, i, j.$$

(Implicerar fullständig graf.)

Specialfall: Δ TSPb: Symmetrisk kostnadsmatrix som uppfyller den begränsade triangelolikheten:

$$c_{ij} \leq c_{ik} + c_{kj} \text{ för alla bågar som finns.}$$

Δ TSP har samma lösning som Δ TSPr. (Återbesök lönar sig aldrig.)

Till varje TSP med lösning finns ett Δ TSPb som har samma optimala lösning. (Addera en konstant till alla kostnader.)

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

$$\text{då } \sum_{i \in N} x_{ij} = 1 \quad \text{för alla } j$$

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

$$\sum_{j \in N} x_{ij} = 1 \quad \text{för alla } i$$

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

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

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

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

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

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

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

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

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

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

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

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

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

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

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

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

Heltalsformulering av TSP, riktad graf

Variabeldefinition:

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

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

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

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

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

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

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

TSP, oriktad graf

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

TSP, oriktad graf

$$\begin{aligned} \min \quad & \sum_{(i,j) \in B} c_{ij} x_{ij} \\ & \sum_{i \in N} x_{ij} = 2 \quad \text{för alla } j \end{aligned}$$

TSP, oriktad graf

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

TSP, oriktad graf

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

TSP, oriktad graf

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

TSP, oriktad graf

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

TSP, oriktad graf

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

För TSP_{Pr}: Ta bort första bivillkoren.

TSP, oriktad graf

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

För TSP_{Pr}: Ta bort första bivillkoren.

Alternativ (ej för TSP_{Pr}):

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

TSP, oriktad graf

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

För TSP_{Pr}: Ta bort första bivillkoren.

Alternativ (ej för TSP_{Pr}):

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

Relaxation av TSP: 1-träd

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

Relaxation av TSP: 1-träd

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

(Alltså inte ett träd.)

Relaxation av TSP: 1-träd

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

(Alltså inte ett träd.)

Rätt antal bågar.

Relaxation av TSP: 1-träd

Billigaste 1-träd: Ett MST för noderna $\{2, 3, \dots, n\}$, 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.

Relaxation av TSP: 1-träd

Billigaste 1-träd: Ett MST för noderna $\{2, 3, \dots, n\}$, 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.

Relaxation av TSP: 1-träd

Billigaste 1-träd: Ett MST för noderna $\{2, 3, \dots, n\}$, 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 av TSP: 1-träd

Billigaste 1-träd: Ett MST för noderna $\{2, 3, \dots, n\}$, plus de två billigaste bågar 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.

Relaxation av TSP: 1-träd

Billigaste 1-träd: Ett MST för noderna $\{2, 3, \dots, n\}$, plus de två billigaste bågar 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.

Relaxation av TSP: 1-träd

Billigaste 1-träd: Ett MST för noderna $\{2, 3, \dots, n\}$, 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:

Förbättring av 1-träd

Nodstraff:

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

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.

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.

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.

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.

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 Gå till 1.

Förbättring av 1-träd

Nodstraff:

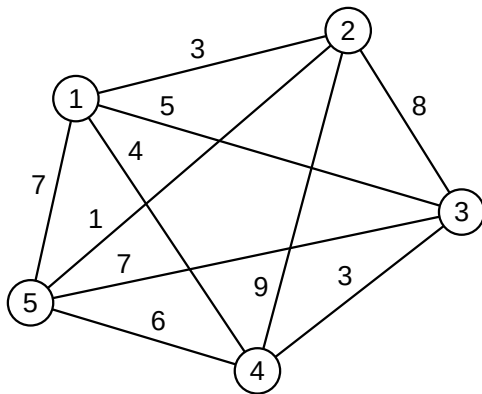
Addition av konstant till c för alla bågar som ansluter till en nod ändrar ej optimal handelsresandetur.

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 Gå till 1.

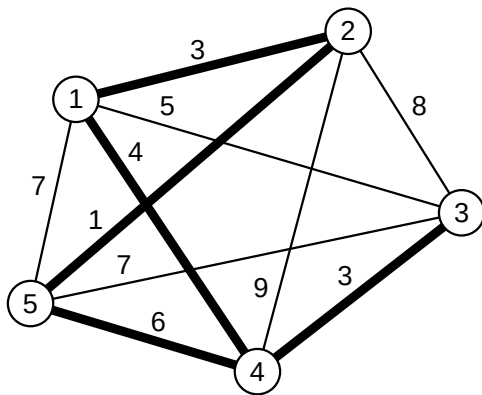
Heuristik: Försök få en tillåten lösning genom att flytta en ändnod för enstaka bågar, från noder med för hög valens till noder med för låg.

Exempel: 1-träd



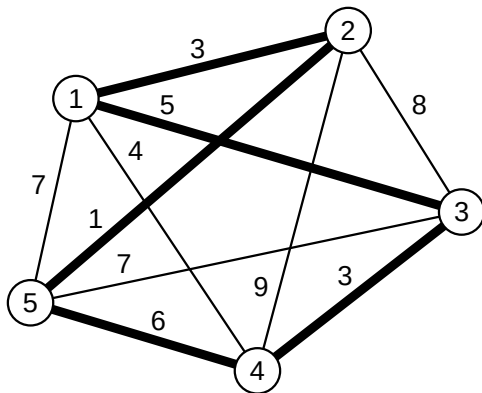
Graf.

Exempel: 1-träd



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

Exempel: 1-träd



Flytta en båge. En handelsresandetur. Kostnad: 18. Övre gräns: 18.
 $17 \leq v^* \leq 18$.

TSP: LP-relaxation

LP-relaxationen samt relaxation av sammanhängandekrav:

TSP: LP-relaxation

LP-relaxationen samt relaxation av sammanhängandekrav:

Kan ge lösningsgraf (där $x_{ij} > 0$) som ej är sammanhängande.

TSP: LP-relaxation

LP-relaxationen samt relaxation av sammanhängandekrav:

Kan ge lösningsgraf (där $x_{ij} > 0$) som ej är sammanhängande.

Lägg till det bivillkor som ej uppfylls.

TSP: LP-relaxation

LP-relaxationen samt relaxation av sammanhängandekrav:

Kan ge lösningsgraf (där $x_{ij} > 0$) som ej är sammanhängande.

Lägg till det bivillkor som ej uppfylls.

- 1 Lös LP-relaxationen.

TSP: LP-relaxation

LP-relaxationen samt relaxation av sammanhängandekrav:

Kan ge lösningsgraf (där $x_{ij} > 0$) som ej är sammanhängande.

Lägg till det bivillkor som ej uppfylls.

- 1 Lös LP-relaxationen.
- 2 Identifiera en nodmängd S som ej är sammanhängande med övriga noder.

TSP: LP-relaxation

LP-relaxationen samt relaxation av sammanhängandekrav:

Kan ge lösningsgraf (där $x_{ij} > 0$) som ej är sammanhängande.

Lägg till det bivillkor som ej uppfylls.

- 1 Lös LP-relaxationen.
- 2 Identifiera en nodmängd S som ej är sammanhängande med övriga noder.
- 3 Lägg till bivillkoret att minst två bågar ska gå ut från S och lös om LP-relaxationen.

TSP: LP-relaxation

LP-relaxationen samt relaxation av sammanhängandekrav:

Kan ge lösningsgraf (där $x_{ij} > 0$) som ej är sammanhängande.

Lägg till det bivillkor som ej uppfylls.

- 1 Lös LP-relaxationen.
- 2 Identifiera en nodmängd S som ej är sammanhängande med övriga noder.
- 3 Lägg till bivillkoret att minst två bågar ska gå ut från S och lös om LP-relaxationen.
- 4 Om lösningen inte är sammanhängande, gå till 2.

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$
1987	532	Padberg och Rinaldi	$1.5 * 10^{1218}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$
1987	532	Padberg och Rinaldi	$1.5 * 10^{1218}$
1987	666	Grötschel och Holland	$1.5 * 10^{1590}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$
1987	532	Padberg och Rinaldi	$1.5 * 10^{1218}$
1987	666	Grötschel och Holland	$1.5 * 10^{1590}$
1987	2392	Padberg och Rinaldi	$7.5 * 10^{7041}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$
1987	532	Padberg och Rinaldi	$1.5 * 10^{1218}$
1987	666	Grötschel och Holland	$1.5 * 10^{1590}$
1987	2392	Padberg och Rinaldi	$7.5 * 10^{7041}$
1994	7397	Applegate, Bixby, Chvatal och Cook	$2.5 * 10^{25405}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$
1987	532	Padberg och Rinaldi	$1.5 * 10^{1218}$
1987	666	Grötschel och Holland	$1.5 * 10^{1590}$
1987	2392	Padberg och Rinaldi	$7.5 * 10^{7041}$
1994	7397	Applegate, Bixby, Chvatal och Cook	$2.5 * 10^{25405}$
1998	13509	Applegate, Bixby, Chvatal och Cook	$1.1 * 10^{49932}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$
1987	532	Padberg och Rinaldi	$1.5 * 10^{1218}$
1987	666	Grötschel och Holland	$1.5 * 10^{1590}$
1987	2392	Padberg och Rinaldi	$7.5 * 10^{7041}$
1994	7397	Applegate, Bixby, Chvatal och Cook	$2.5 * 10^{25405}$
1998	13509	Applegate, Bixby, Chvatal och Cook	$1.1 * 10^{49932}$
2001	15112	Applegate, Bixby, Chvatal och Cook	$1.4 * 10^{56593}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$
1987	532	Padberg och Rinaldi	$1.5 * 10^{1218}$
1987	666	Grötschel och Holland	$1.5 * 10^{1590}$
1987	2392	Padberg och Rinaldi	$7.5 * 10^{7041}$
1994	7397	Applegate, Bixby, Chvatal och Cook	$2.5 * 10^{25405}$
1998	13509	Applegate, Bixby, Chvatal och Cook	$1.1 * 10^{49932}$
2001	15112	Applegate, Bixby, Chvatal och Cook	$1.4 * 10^{56593}$
2004	24978	Applegate, Bixby, Chvatal och Cook	$3.9 * 10^{98992}$

TSP: De största som lösts till optimalitet

År	n	Forskare	Turer
1954	49	Dantzig, Fulkerson och Johnson	$1.2 * 10^{61}$
1971	64	Held och Karp	$2.0 * 10^{87}$
1975	100	Camerini, Fratta och Maffioli	$9.3 * 10^{155}$
1977	120	Grötschel	$5.6 * 10^{196}$
1980	318	Crowder och Padberg	$2.1 * 10^{659}$
1987	532	Padberg och Rinaldi	$1.5 * 10^{1218}$
1987	666	Grötschel och Holland	$1.5 * 10^{1590}$
1987	2392	Padberg och Rinaldi	$7.5 * 10^{7041}$
1994	7397	Applegate, Bixby, Chvatal och Cook	$2.5 * 10^{25405}$
1998	13509	Applegate, Bixby, Chvatal och Cook	$1.1 * 10^{49932}$
2001	15112	Applegate, Bixby, Chvatal och Cook	$1.4 * 10^{56593}$
2004	24978	Applegate, Bixby, Chvatal och Cook	$3.9 * 10^{98992}$
2006	85900	Applegate, Bixby, Chvatal, Cook, Espinoza, Goycoolea och Helsgaun	$1.1 * 10^{386522}$

TSP: 49 noder

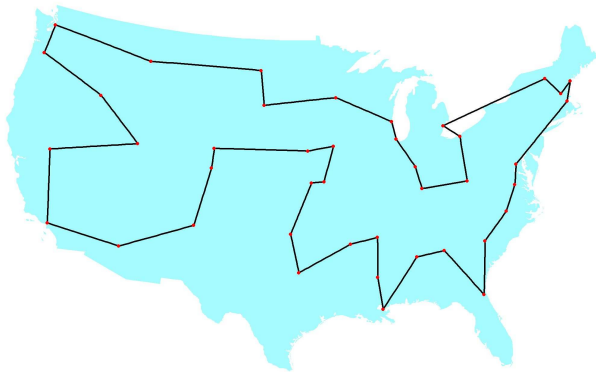


Bild: <http://www.tsp.gatech.edu>

TSP: 532 noder

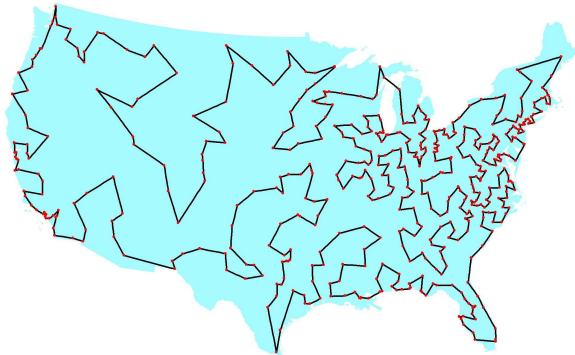


Bild: <http://www.tsp.gatech.edu>

TSP: 666 noder

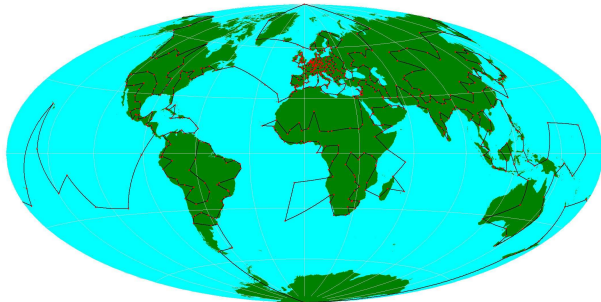


Bild: <http://www.tsp.gatech.edu>

TSP: 15112 noder

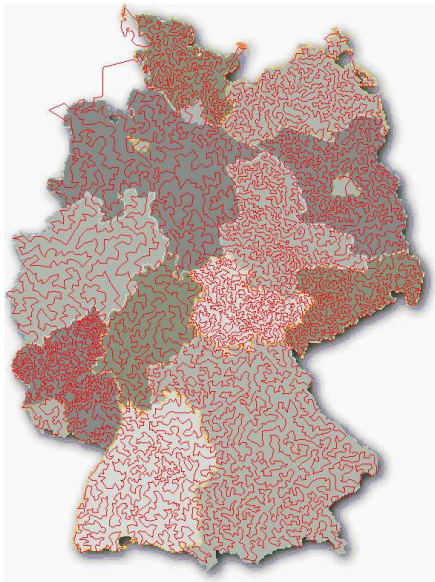


Bild: <http://www.tsp.gatech.edu>

TSP: 24978 noder



Bild: <http://www.tsp.gatech.edu>

TSP: Sverige, optimallosningen

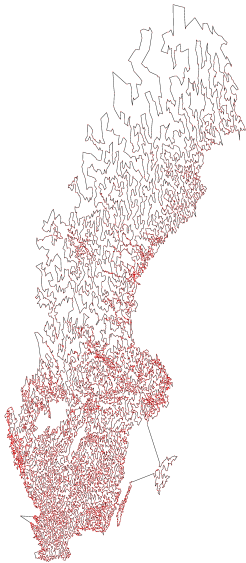


Bild: <http://www.tsp.gatech.edu>

TSP: Sverige, optimallosningen

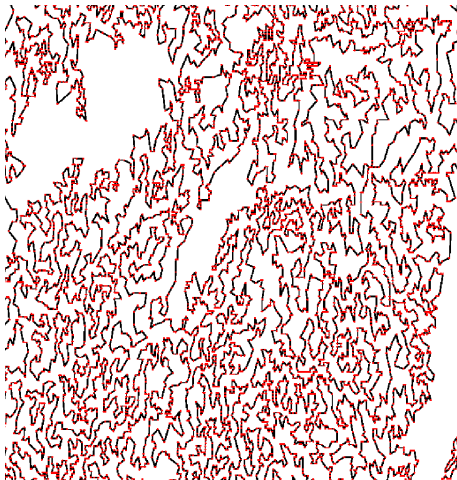


Bild: <http://www.tsp.gatech.edu>

Tur: 85900 noder

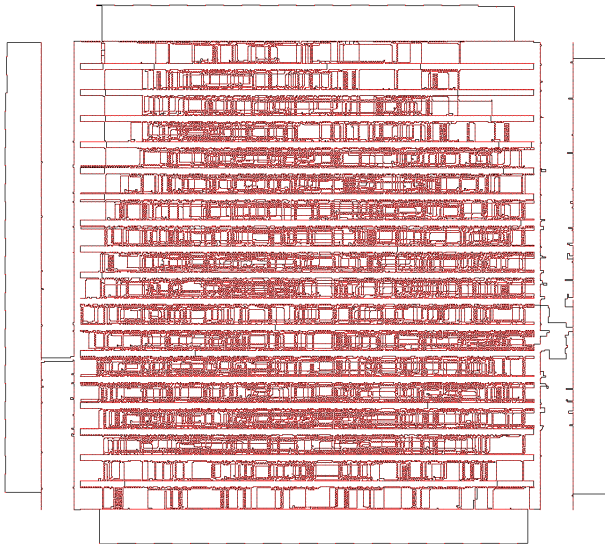


Bild: <http://www.tsp.gatech.edu>

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683		

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683	855 331	0.041%
010826			

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683		
010826		855 331	0.041%
010904	855 618		0.034%

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683	855 331	0.041%
010826			
010904	855 618		
010920	855 612		

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683	855 331	0.041%
010826			
010904	855 618		
010920	855 612		
010930	855 610		

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683	855 331	0.041%
010826			
010904	855 618		0.034%
010920	855 612		0.033%
010930	855 610		0.033%
030316	855 602		0.032%

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683	855 331	0.041%
010826			
010904	855 618		0.034%
010920	855 612		0.033%
010930	855 610		0.033%
030316	855 602		0.032%
030318	855 597		0.031%

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683	855 331	0.041%
010826			
010904	855 618		
010920	855 612		
010930	855 610		
030316	855 602	855 493	0.032%
030318	855 597		
030324			

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683	855 331	0.041%
010826			
010904	855 618		
010920	855 612		
010930	855 610		
030316	855 602	855 493	0.032%
030318	855 597		0.031%
030324			0.012%
030602		855 528	0.009%

Framsteg med Sverige-exemplet

Datum	Tur (Övre gräns)	Undre gräns	Relativt gap
010824	855 683	855 331	0.041%
010826			
010904	855 618		
010920	855 612		
010930	855 610		
030316	855 602	855 493	0.032%
030318	855 597		
030324			
030602			
040520			
		855 528	0.009%
		855 597	0

Tillåtna lösningar till 85900 noders-exemplet

Datum	Kostnad	Vem
910607	142 514 146	David Johnson
960329	142 487 006	Concorde
970923	142 482 068	Concorde
981014	142 416 327	Keld Helsgaun
991022	142 409 553	Concorde
010618	142 406 493	Keld Helsgaun
010627	142 405 532	Keld Helsgaun
010831	142 395 130	Concorde
011214	142 393 738	Keld Helsgaun
020915	142 385 237	Hisao Tamaki
021212	142 383 704	Keld Helsgaun
030319	142 383 467	Nguyen Dinh Hung
030428	142 383 189	Keld Helsgaun
031223	142 383 011	Keld Helsgaun
040502	142 382 641	Keld Helsgaun

Steinerträdsproblemet

Givet: Graf $G = (N, E)$, en nodmängd $N' \subset N$ och bågkostnader c .

Steinerträdsproblemet

Givet: Graf $G = (N, E)$, en nodmängd $N' \subset N$ och bågkostnader c .

Ett Steinerträd *måste* omfatta samtliga noder i N' men *får* också omfatta noder i $N \setminus N'$.

Steinerträdsproblemet

Givet: Graf $G = (N, E)$, en nodmängd $N' \subset N$ och bågkostnader c .

Ett Steinerträd *måste* omfatta samtliga noder i N' men *får* också omfatta noder i $N \setminus N'$.

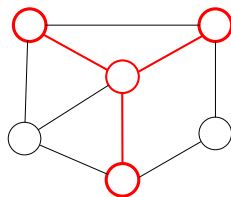
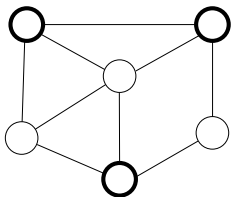
Steinerträdsproblemet: Finn ett Steinerträd med minimal kostnad.

Steinerträdsproblemet

Givet: Graf $G = (N, E)$, en nodmängd $N' \subset N$ och bågkostnader c .

Ett Steinerträd *måste* omfatta samtliga noder i N' men *får* också omfatta noder i $N \setminus N'$.

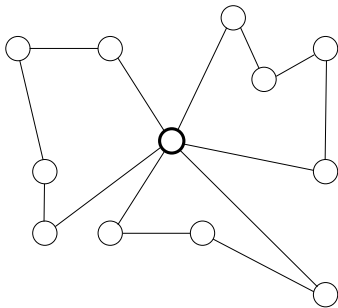
Steinerträdsproblemet: Finn ett Steinerträd med minimal kostnad.



Mittennoden behöver inte vara med, men det kan bli billigare att ta med den.

Ruttplaneringsproblemet

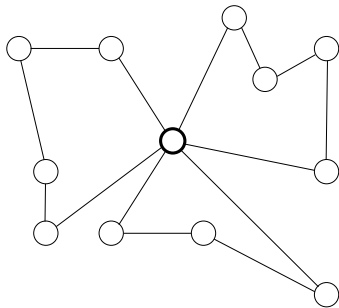
Ett antal lastbilar skall köra ut varor till ett antal kunder. Varje lastbil startar i en depå, kör runt till några kunder och levererar varor och återvänder till depån. Lastbilarna har begränsad lastkapacitet och kunderna given efterfrågan.



Varje lastbil kör en handelsresandetur i en mängd noder som ska bestämmas. Samtliga noder skall täckas av någon tur.

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.

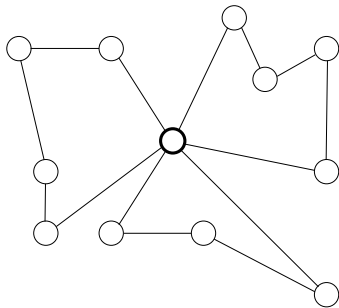


Varje lastbil kör en handelsresandetur i en mängd noder som ska bestämmas. Samtliga noder skall täckas av någon tur.

Målfunktion: Minimera kostnaden

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.

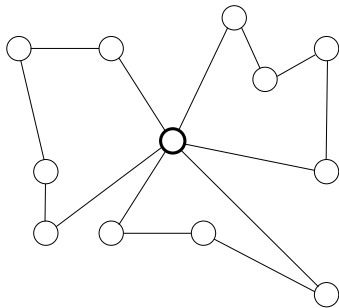


Varje lastbil kör en handelsresandetur i en mängd noder som ska bestämmas. Samtliga noder skall täckas av någon tur.

Målfunktion: Minimera kostnaden eller avgasutsläppen

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.

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.

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.

Matching: högst en av bågarna ansluter till någon nod.

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.

Matching: högst en av bågarna ansluter till någon nod.

$$\max \sum_i \sum_j c_{ij} x_{ij}$$

$$\text{då} \quad \sum_j (x_{ij} + x_{ji}) \leq 1 \quad \text{för alla } i$$

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

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.

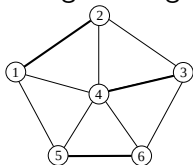
Matching: högst en av bågarna ansluter till någon nod.

$$\max \sum_i \sum_j c_{ij} x_{ij}$$

$$\text{då} \quad \sum_j (x_{ij} + x_{ji}) \leq 1 \quad \text{för alla } i$$

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

Perfekt matchning: samtliga noder har valens ett.



Matchningsproblemet

Alternnerande väg: Varannan båge ingår i matchningen.

Matchningsproblemet

Alternnerande väg: Varannan båge ingår i matchningen.

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

Matchningsproblemet

Alternerande väg: Varannan båge ingår i matchningen.

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

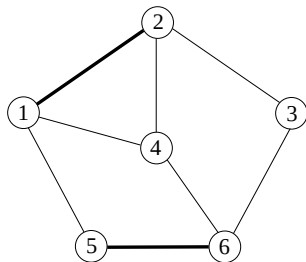
Metod: Byt matchad mot omatchad längs en utökande väg.

Matchningsproblemet

Alternerande väg: Varannan båge ingår i matchningen.

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

Metod: Byt matchad mot omatchad längs en utökande väg.

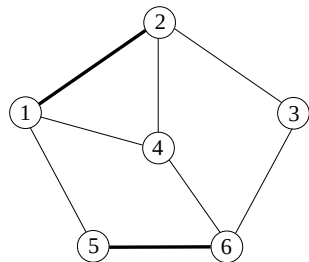


Matchningsproblemet

Alternerande väg: Varannan båge ingår i matchningen.

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

Metod: Byt matchad mot omatchad längs en utökande väg.



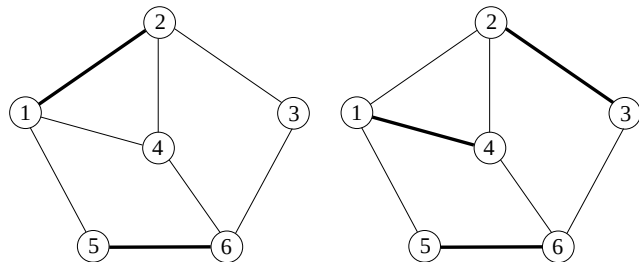
Utökande väg: 4 - 1 - 2 - 3, ger maximal matchning.

Matchningsproblemet

Alternerande väg: Varannan båge ingår i matchningen.

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

Metod: Byt matchad mot omatchad längs en utökande väg.



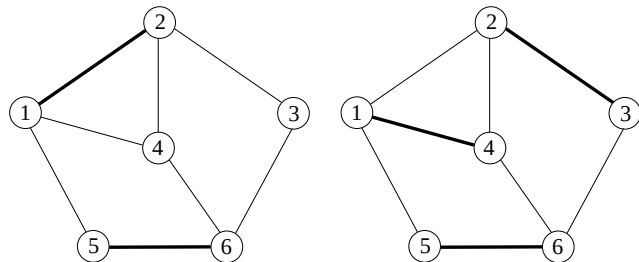
Utökande väg: 4 - 1 - 2 - 3, ger maximal matchning.

Matchningsproblemet

Alternerande väg: Varannan båge ingår i matchningen.

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

Metod: Byt matchad mot omatchad längs en utökande väg.



Utökande väg: 4 - 1 - 2 - 3, ger maximal matchning.

Sats (Berge 1957)

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

Kinesiska brevbärarproblemet

Kinesiska brevbärrarproblemet

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

Kinesiska brevbärrarproblemet

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

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

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 *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.



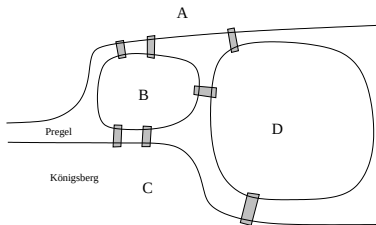
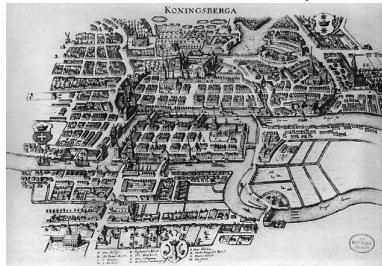
Kinesiska brevbärarproblemet

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

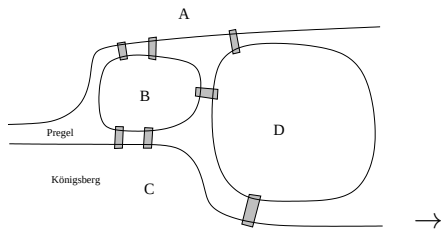
Det *kinesiska brevbärarproblemet* är att finna en brevbärartur med minimal kostnad.

Königsbergs broar: (Euler, 1700-talet)

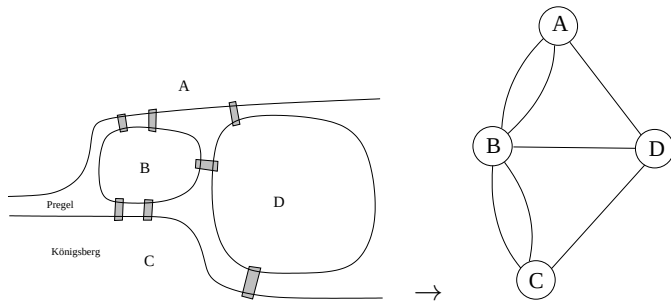
Försök finna en tur som passerar varje bro en gång.



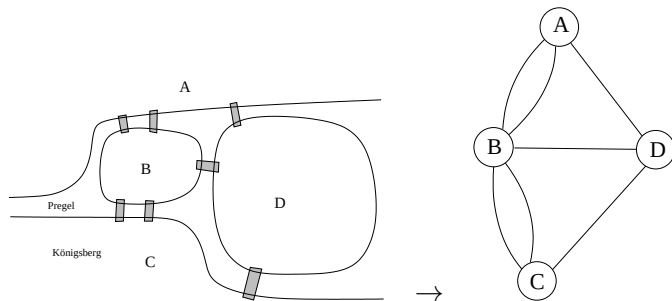
Kinesiska brevbärarproblemet



Kinesiska brevbärarproblemet

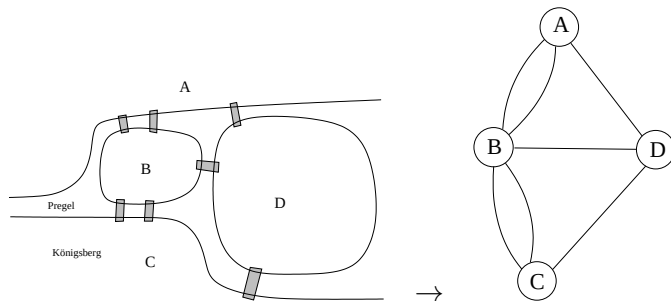


Kinesiska brevbärarproblemet



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

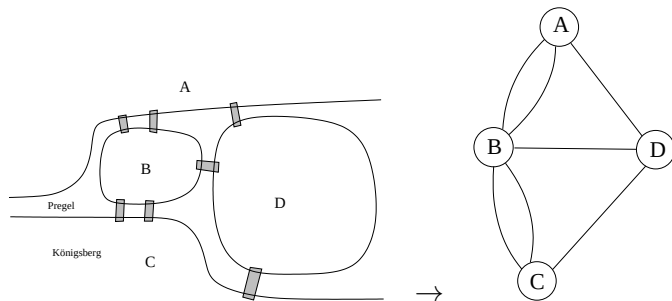
Kinesiska brevbärarproblemet



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

En Eulercykel är en optimal brevbärartur, om den finns.

Kinesiska brevbärarproblemet

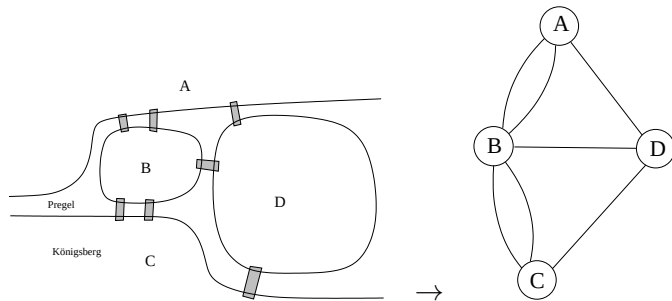


En **Eulercykel** är en cykel som använder varje *båge* i grafen exakt en gång.

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.

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 brevbärrarproblemet: Modell

Matematisk modell:

Kinesiska brevbärarproblemet: Modell

Matematisk modell:

x_{ij} : antalet gånger båge (i, j) trafikeras.

z_i : antal gånger nod i passeras i turen.

Kinesiska brevbärarproblemet: Modell

Matematisk modell:

x_{ij} : antalet gånger båge (i, j) trafikeras.

z_i : antal gånger nod i passeras i turen.

$$\min \sum_i \sum_j c_{ij} x_{ij}$$

$$\text{då} \quad \sum_j (x_{ij} + x_{ji}) - 2z_i = 0 \quad \text{för alla } i$$

$$x_{ij} \geq 1, \text{ heltal} \quad \text{för alla } i, j$$

$$z_i \geq 1, \text{ heltal} \quad \text{för alla } i$$

Kinesiska brevbärarproblemet

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

Kinesiska brevbärarproblemet

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

Metod:

Minimera extraarbetet.

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.

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.

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 vidare.

Lägg till bågar mellan noder med udda valens, så att alla noder får jämn valens,

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 vidare.

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

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 vidare.

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.

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 vidare.

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).

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.

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 vidare.

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

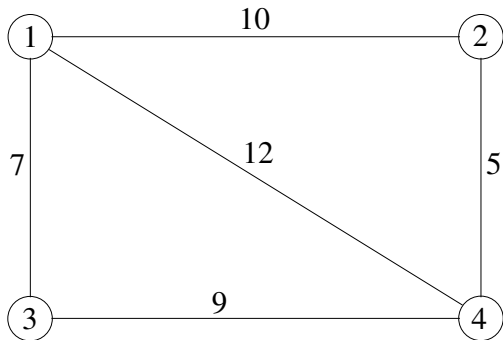
En Eulercykel ger då den billigaste brevbärarturen.

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

2. Finn Eulercykeln.

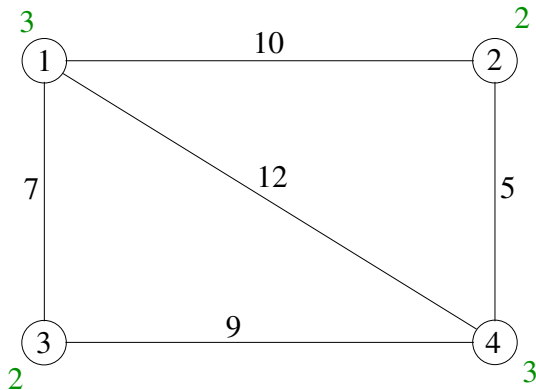
Polynomisk optimerande metod.

Kinesiska brevbärarproblemet: Exempel



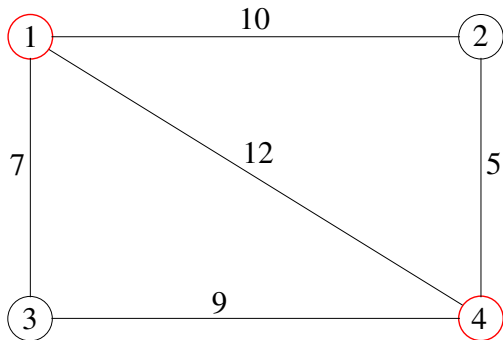
En graf.

Kinesiska brevbärarproblemet: Exempel



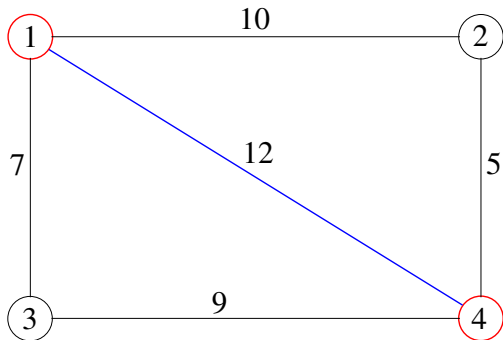
Räkna nodvalens.

Kinesiska brevbärarproblemet: Exempel



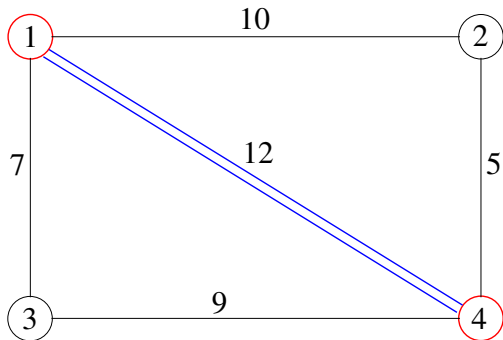
Noder med udda valens.

Kinesiska brevbärarproblemet: Exempel



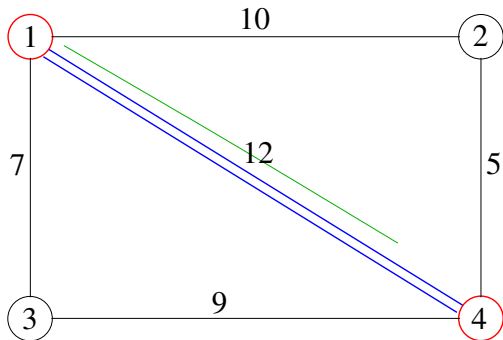
Finn billigaste matchning.

Kinesiska brevbärarproblemet: Exempel



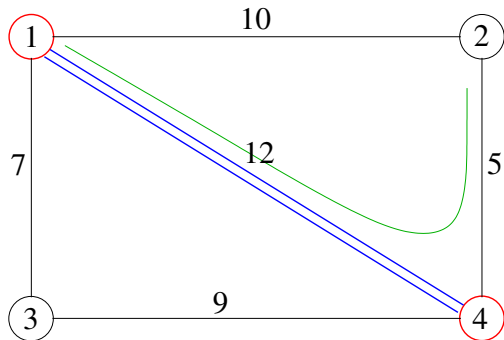
Dubblera bågar.

Kinesiska brevbärarproblemet: Exempel



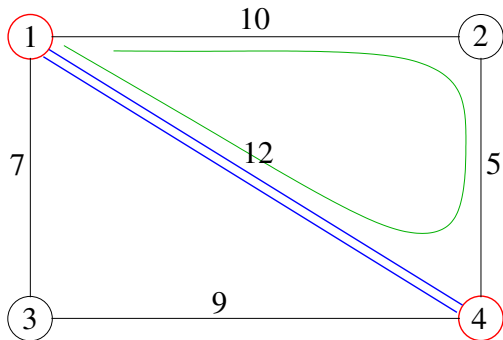
Finn Eulercykel.

Kinesiska brevbärarproblemet: Exempel



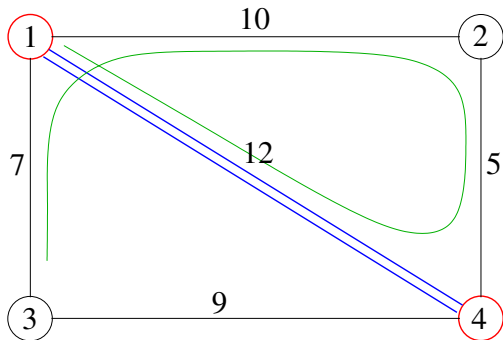
Finn Eulercykel.

Kinesiska brevbärarproblemet: Exempel



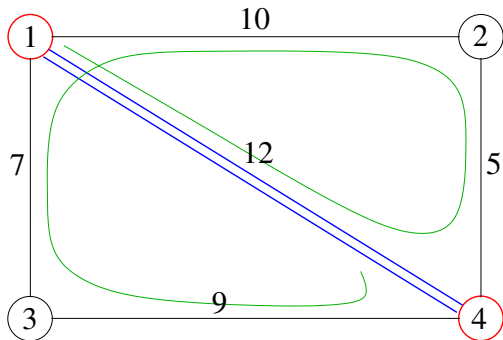
Finn Eulercykel.

Kinesiska brevbärarproblemet: Exempel



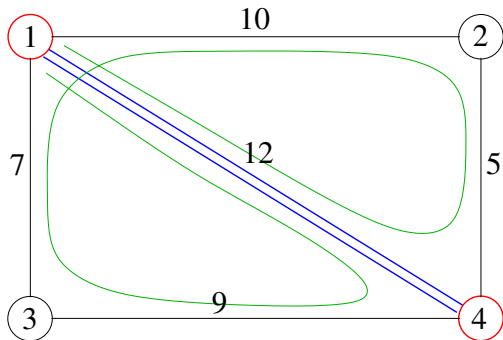
Finn Eulercykel.

Kinesiska brevbärarproblemet: Exempel



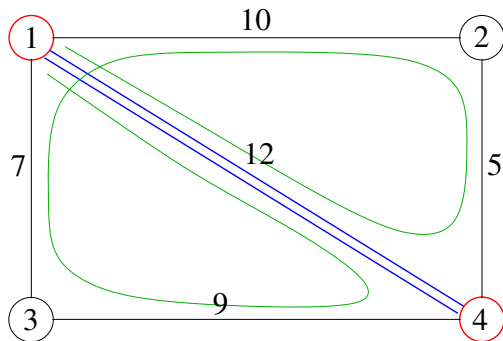
Finn Eulercykel.

Kinesiska brevbärarproblemet: Exempel



Finn Eulercykel.

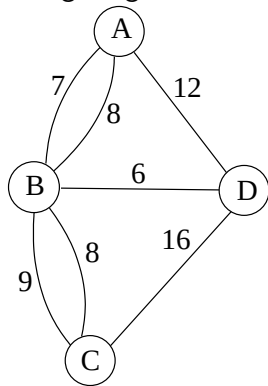
Kinesiska brevbärarproblemet: Exempel



Finn Eulercykel. Kostnad: $43 + 12 = 55$.

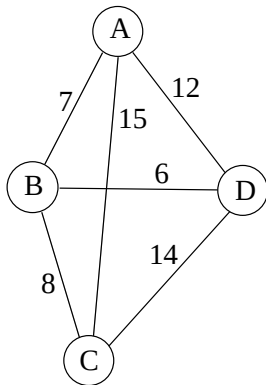
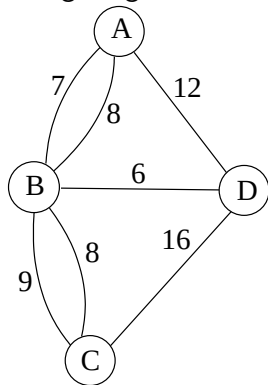
Kinesiska brevbärarproblemet

Königsberg:



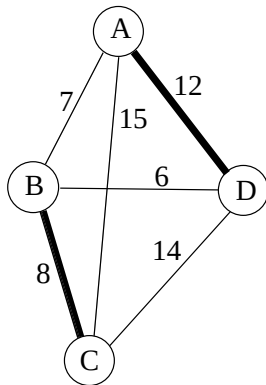
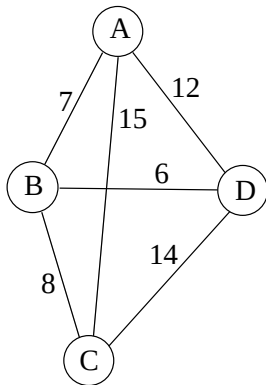
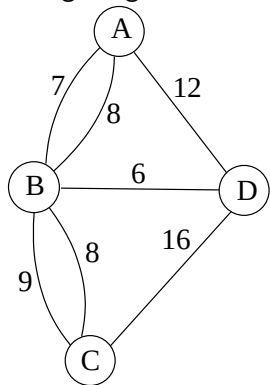
Kinesiska brevbärarproblemet

Königsberg:



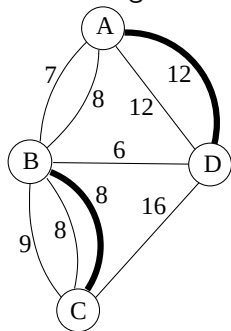
Kinesiska brevbärarproblemet

Königsberg:



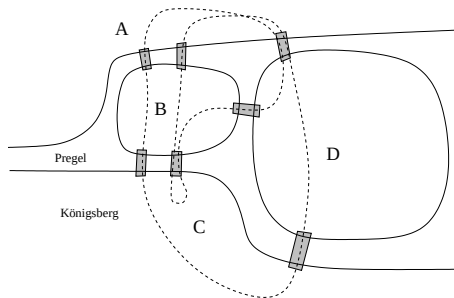
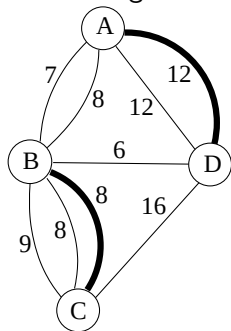
Kinesiska brevbärarproblemet

Addera bågarna i billigaste perfekta matchning. Finn valfri Eulercykel.



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.

Königsberg idag: Kaliningrad

