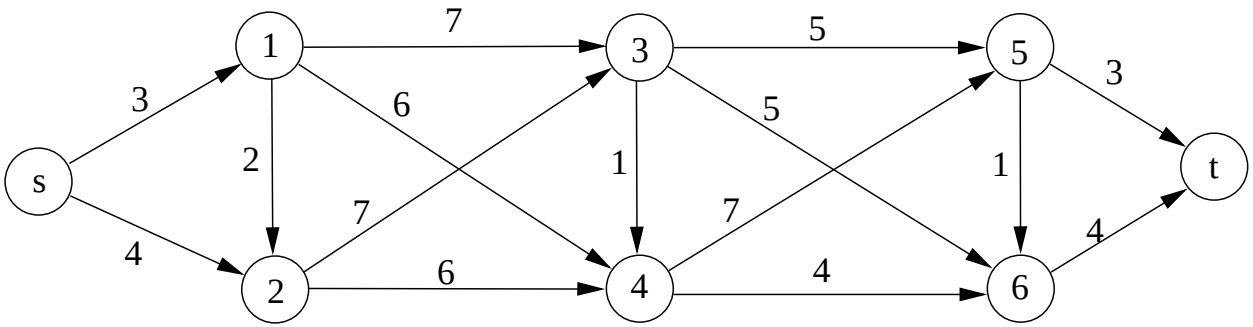
Vägar: Billigaste väg

En student ska cykla från bostaden, s, till tentasalen, t, på kortast möjliga tid (för att inte komma försent).

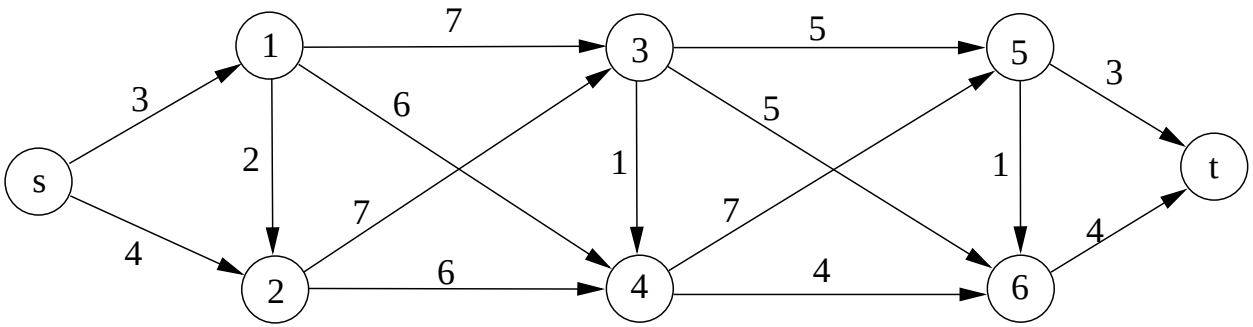
Vägar: Billigaste väg

En student ska cykla från bostaden, s, till tentasalen, t, på kortast möjliga tid (för att inte komma försent).



Vägar: Billigaste väg

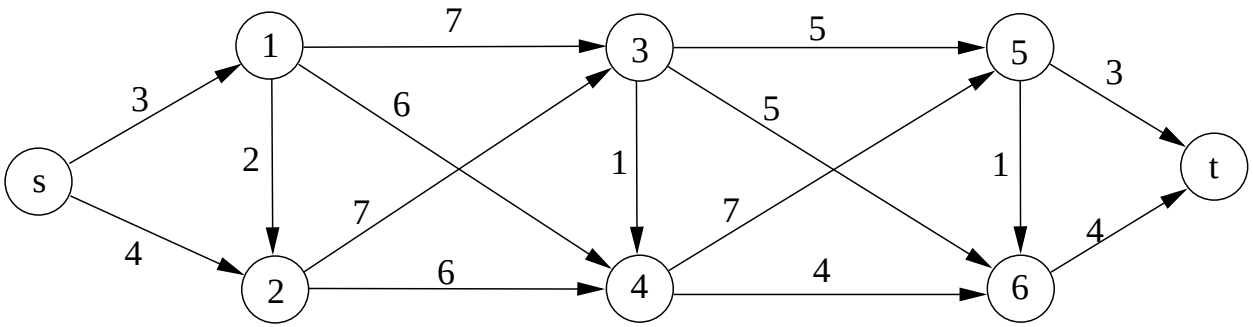
En student ska cykla från bostaden, s, till tentasalen, t, på kortast möjliga tid (för att inte komma försent).



Indata: Riktad graf med bågkostnader c, start/slutnod s, t.

Vägar: Billigaste väg

En student ska cykla från bostaden, s, till tentasalen, t, på kortast möjliga tid (för att inte komma försent).

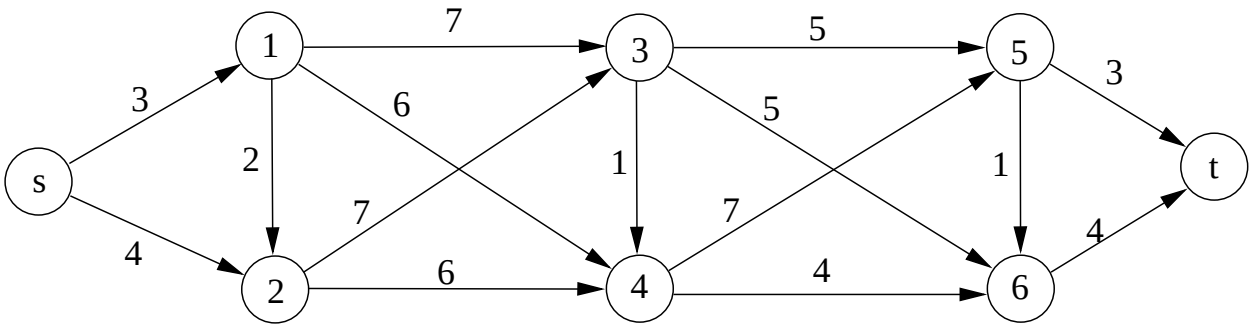


Indata: Riktad graf med bågkostnader c, start/slutnod s, t.

Billigaste väg-problemet: Finn en väg från s till t med minimal kostnad.

Vägar: Billigaste väg

En student ska cykla från bostaden, s, till tentasalen, t, på kortast möjliga tid (för att inte komma försent).



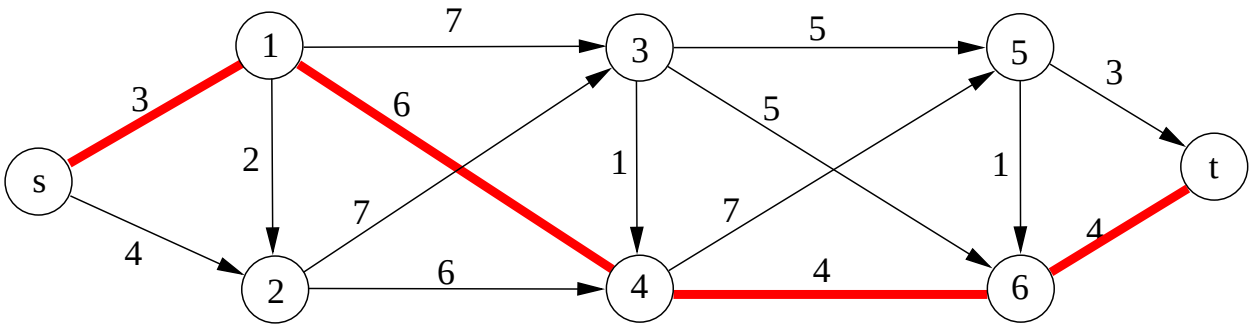
Indata: Riktad graf med bågkostnader c, start/slutnod s, t.

Billigaste väg-problemet: Finn en väg från s till t med minimal kostnad.

Kostnaden för en väg är summan av kostnaderna för de bågar som ingår i vägen.

Vägar: Billigaste väg

En student ska cykla från bostaden, s, till tentasalen, t, på kortast möjliga tid (för att inte komma försent).



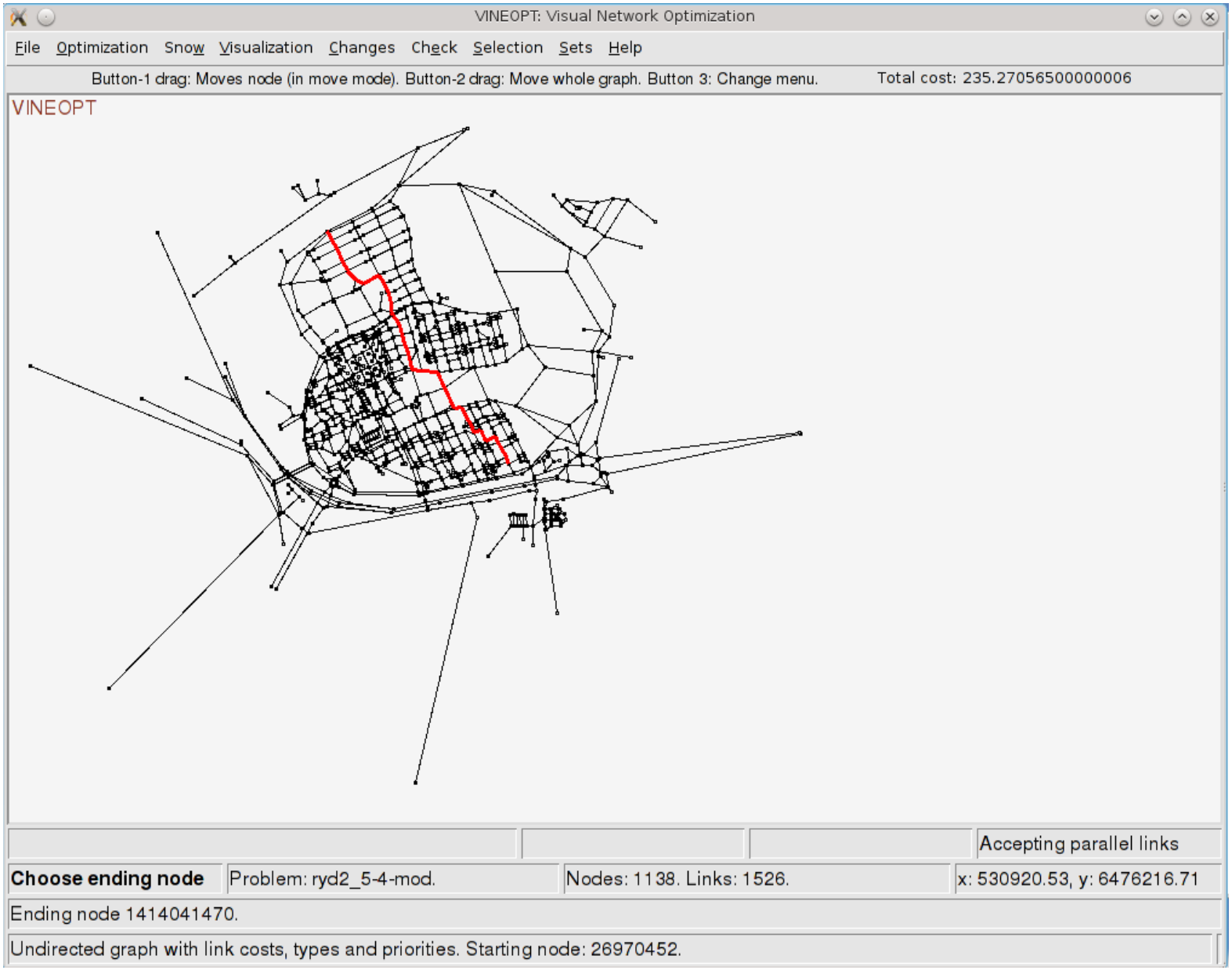
Indata: Riktad graf med bågkostnader c, start/slutnod s, t.

Billigaste väg-problemet: Finn en väg från s till t med minimal kostnad.

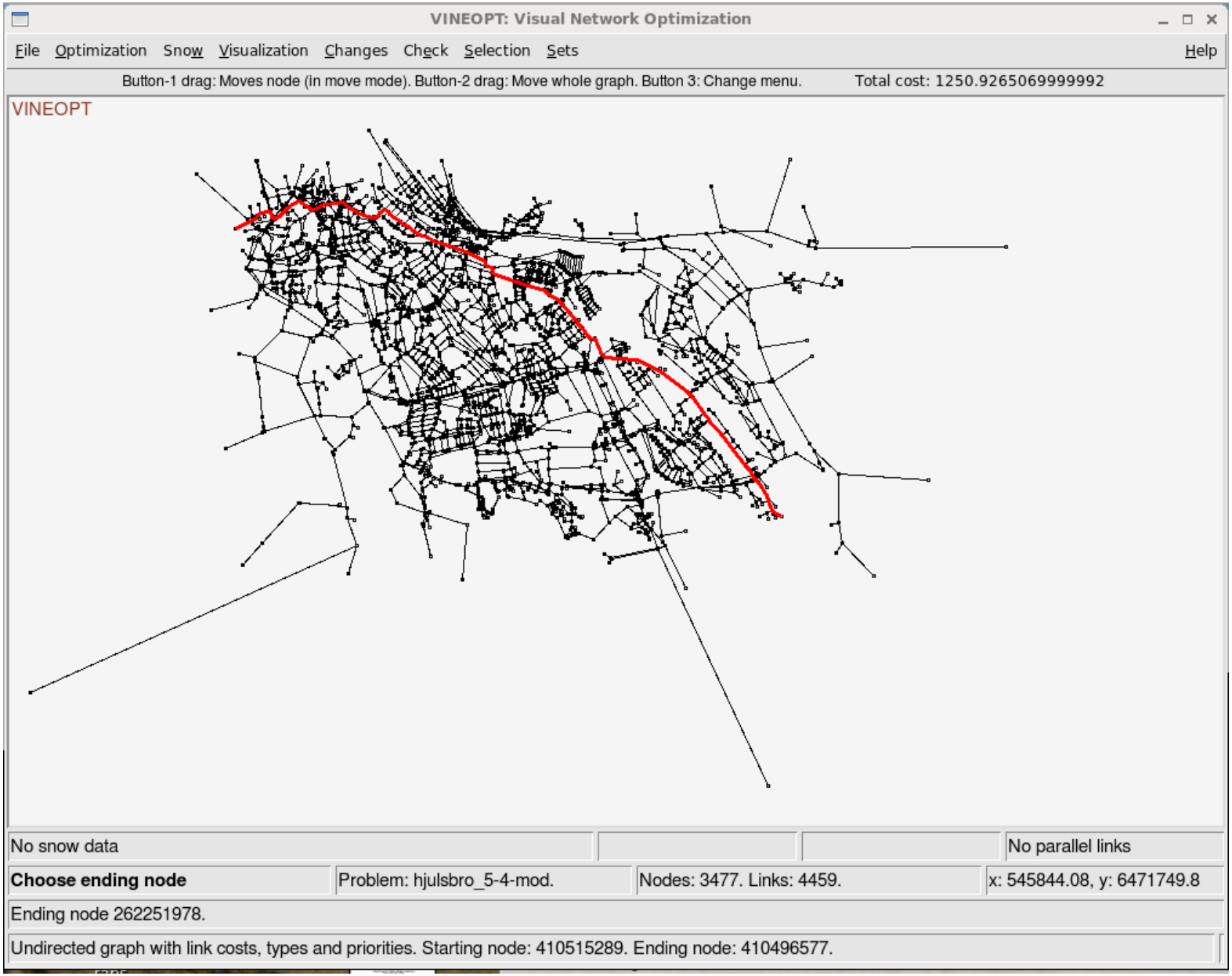
Kostnaden för en väg är summan av kostnaderna för de bågar som ingår i vägen.

Skicka en (odelbar) enhet från s till t på billigaste sätt.

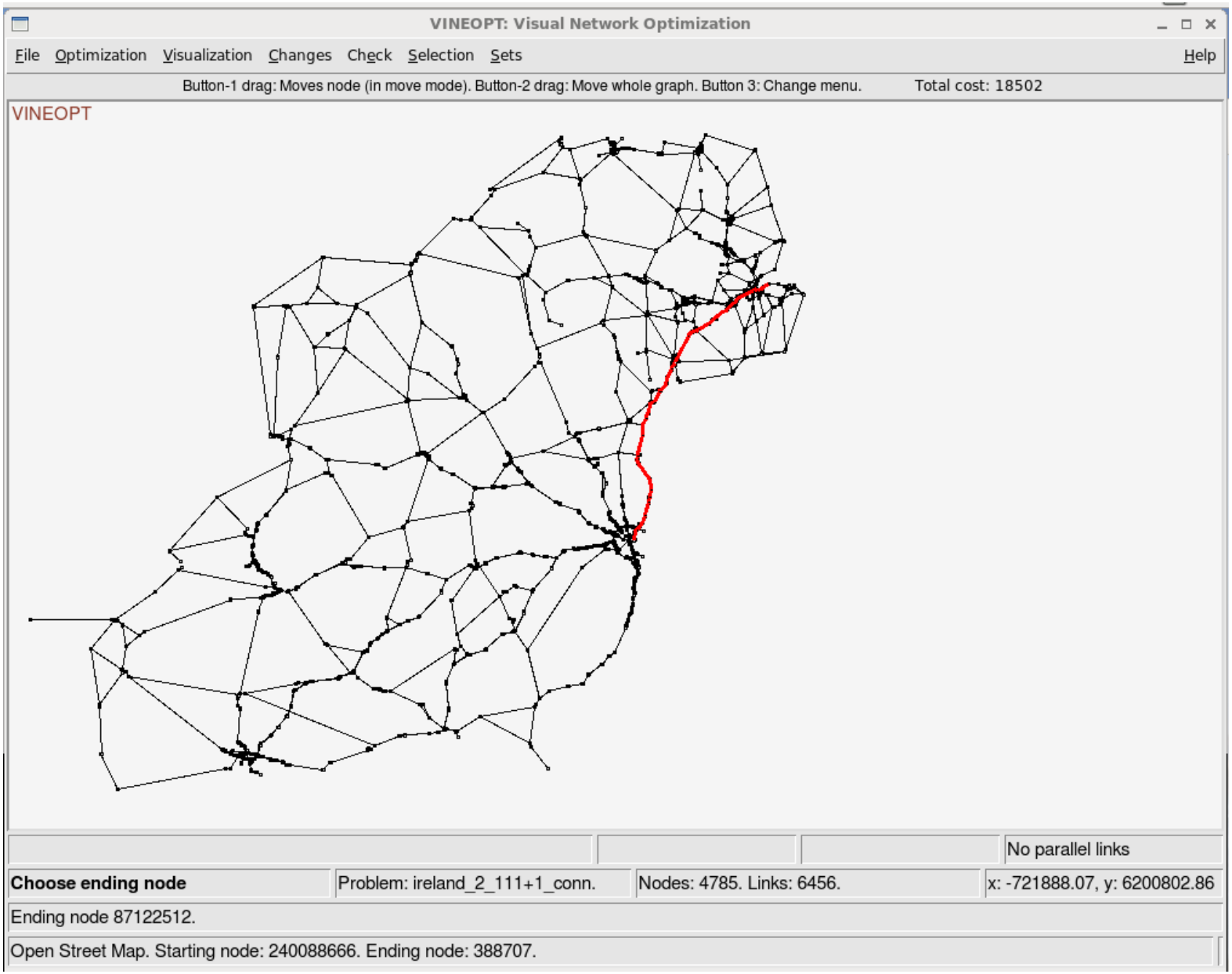
Billigaste väg



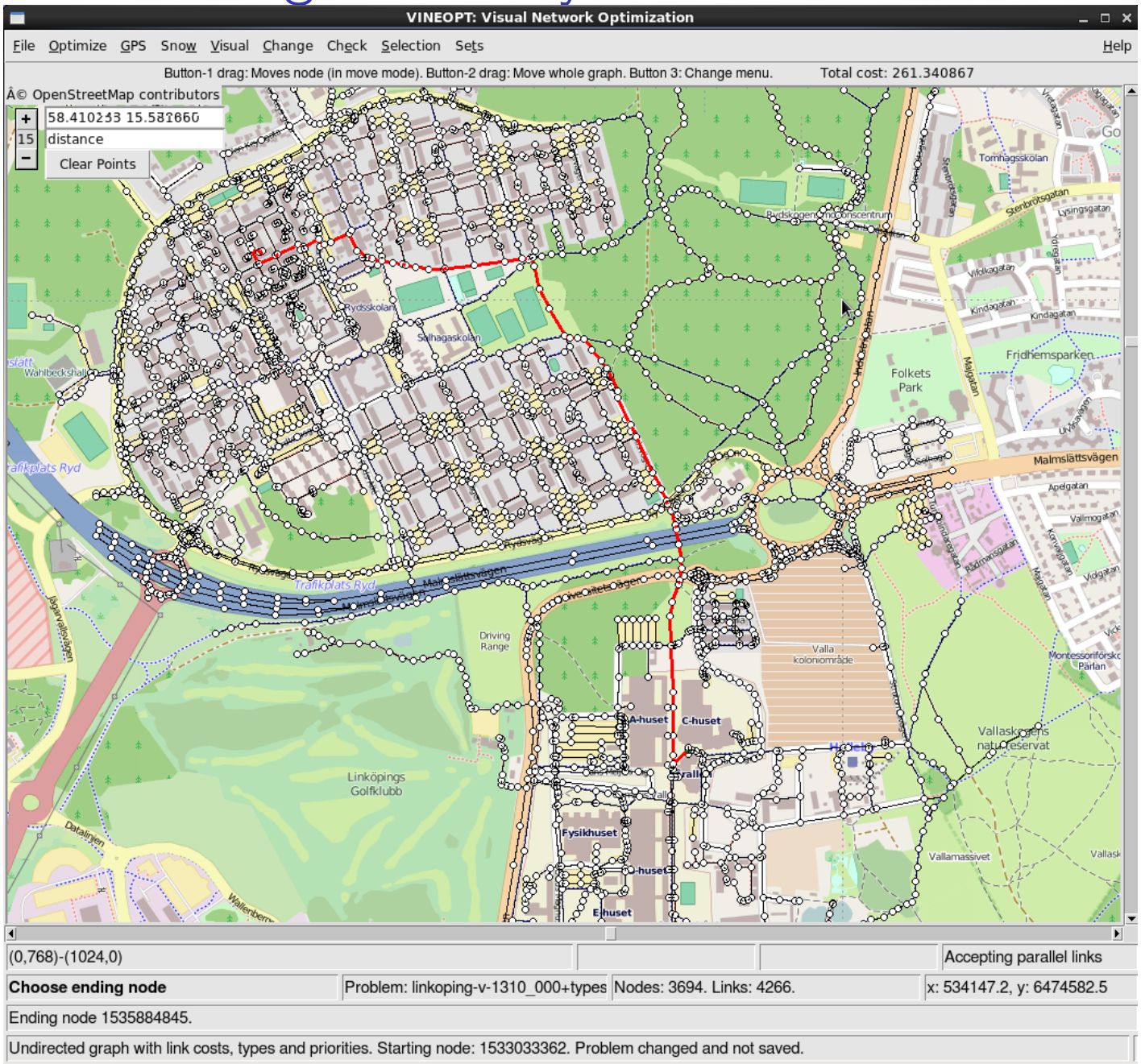
Billigaste väg



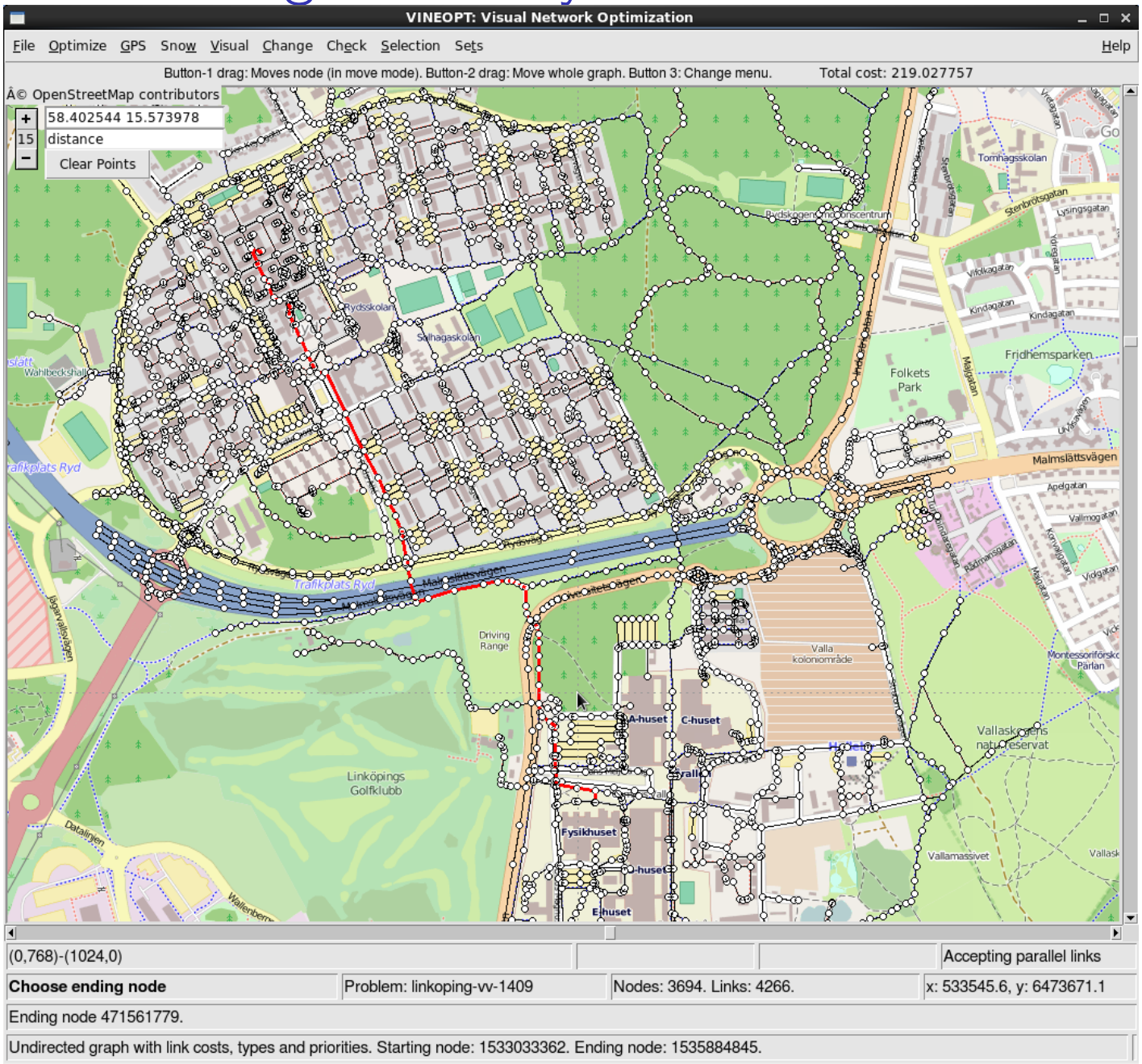
Billigaste väg



Kortaste väg mellan Ryd och LiU



Kortaste väg mellan Ryd och LiU



Billigaste väg: Matematisk modell

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

Billigaste väg: Matematisk modell

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

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

Billigaste väg: Matematisk modell

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

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

$$\text{då} \quad \sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} =$$

Billigaste väg: Matematisk modell

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

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

$$\text{då} \quad \sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases} \quad \text{för alla } i \in N$$

Billigaste väg: Matematisk modell

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

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

$$\text{då} \quad \sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases} \quad \text{för alla } i \in N$$

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

Billigaste väg: Matematisk modell

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

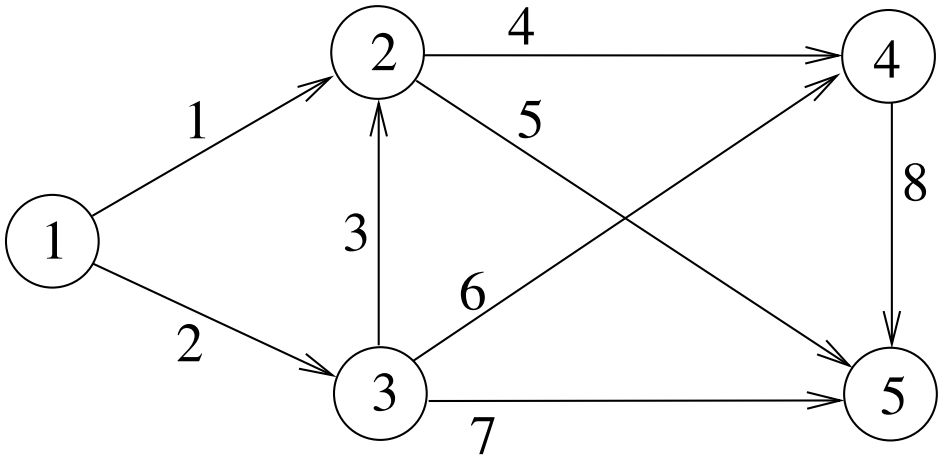
$$\min \quad \sum_{(i,j)\in B} c_{ij} x_{ij}$$

$$\text{då} \quad \sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases} \quad \text{för alla } i \in N$$

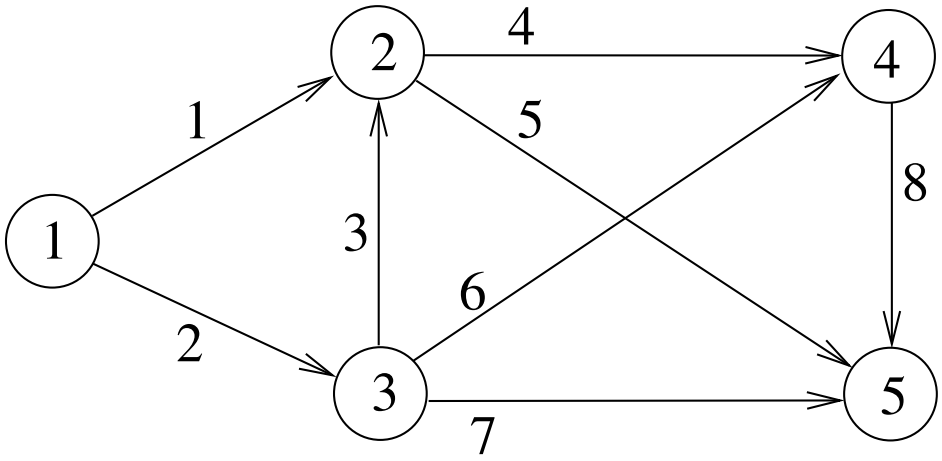
$$x_{ij} \in \{0, 1\} \text{ för alla } (i, j) \in B$$

Nodjämviktsvillkor: (in - ut)

Exempel

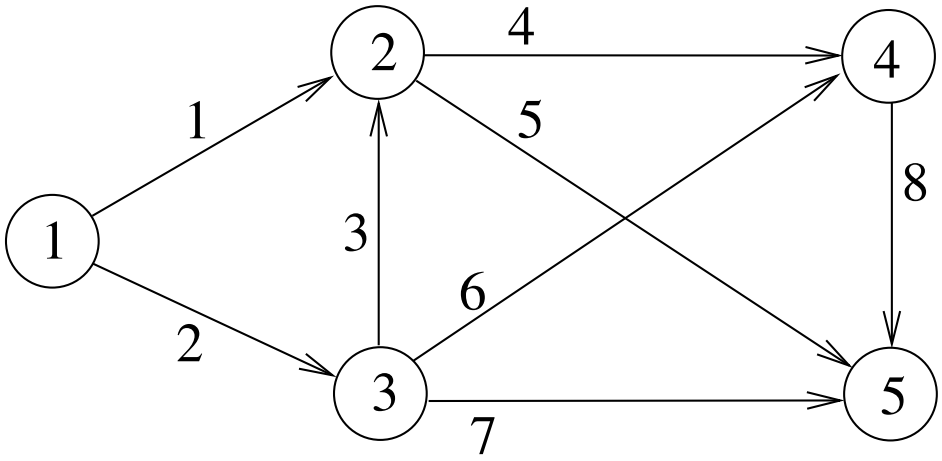


Exempel



Bivillkor:

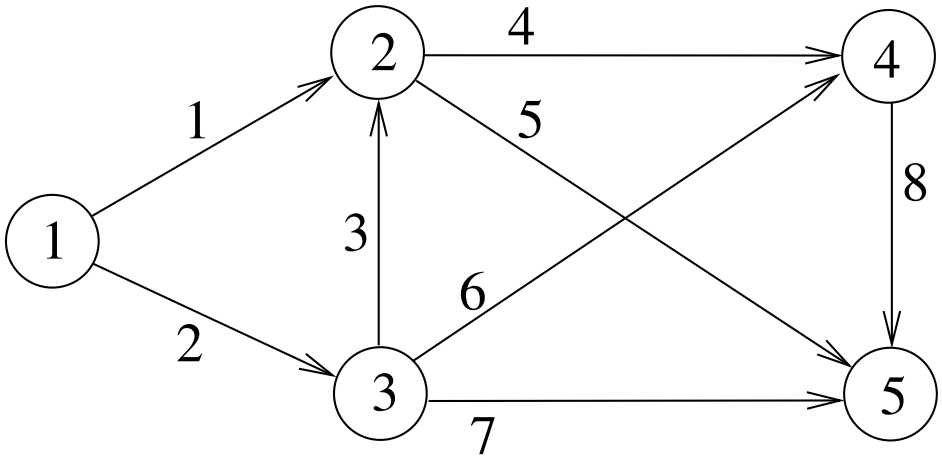
Exempel



Bivillkor:

(in - ut)

Exempel



Bivillkor:

(in - ut)

$$
\begin{array}{rrrrrrrrrrr}
-x_{12} & -x_{13} & & & & & & & = & -1 & (1)\\
x_{12} & & +x_{32} & -x_{24} & -x_{25} & & & & = & 0 & (2)\\
& x_{13} & -x_{32} & & & -x_{34} & -x_{35} & & = & 0 & (3)\\
& & & x_{24} & & +x_{34} & & -x_{45} & = & 0 & (4)\\
& & & & x_{25} & & +x_{35} & +x_{45} & = & 1 & (5)
\end{array}
$$

Billigaste väg: Matematisk modell i vektor/matrisform

Med x som bågvektor:

Billigaste väg: Matematisk modell i vektor/matrisform

Med x som bågvektor:

$$\begin{aligned} \min \quad & c^T x \\ \text{då} \quad & Ax = b \\ & x \in \{0,1\}^m \end{aligned}$$

Billigaste väg: Matematisk modell i vektor/matrisform

Med x som bågvektor:

$$\begin{aligned} \min \quad & c^T x \\ \text{då} \quad & Ax = b \\ & x \in \{0,1\}^m \end{aligned}$$

$$b_i = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases}$$

Billigaste väg: Matematisk modell i vektor/matrisform

Med x som bågvektor:

$$\begin{aligned} \min \quad & c^T x \\ \text{då} \quad & Ax = b \\ & x \in \{0,1\}^m \end{aligned}$$

$$b_i = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases}$$

$$a_{ik} = \begin{cases} -1 & \text{om båge } k \text{ startar i nod } i \\ 1 & \text{om båge } k \text{ slutar i nod } i \\ 0 & \text{för övrigt} \end{cases}$$

Billigaste väg: Matematisk modell i vektor/matrisform

Med x som bågvektor:

$$\begin{array}{ll} \min & c^T x \\ \text{då} & Ax = b \\ & x \in \{0,1\}^m \end{array}$$

$$b_i = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases}$$

$$a_{ik} = \begin{cases} -1 & \text{om båge } k \text{ startar i nod } i \\ 1 & \text{om båge } k \text{ slutar i nod } i \\ 0 & \text{för övrigt} \end{cases}$$

A kallas anslutningsmatris.

Billigaste väg: Matematisk modell i vektor/matrisform

Med x som bågvektor:

$$\begin{aligned} \min \quad & c^T x \\ \text{då} \quad & Ax = b \\ & x \in \{0,1\}^m \end{aligned}$$

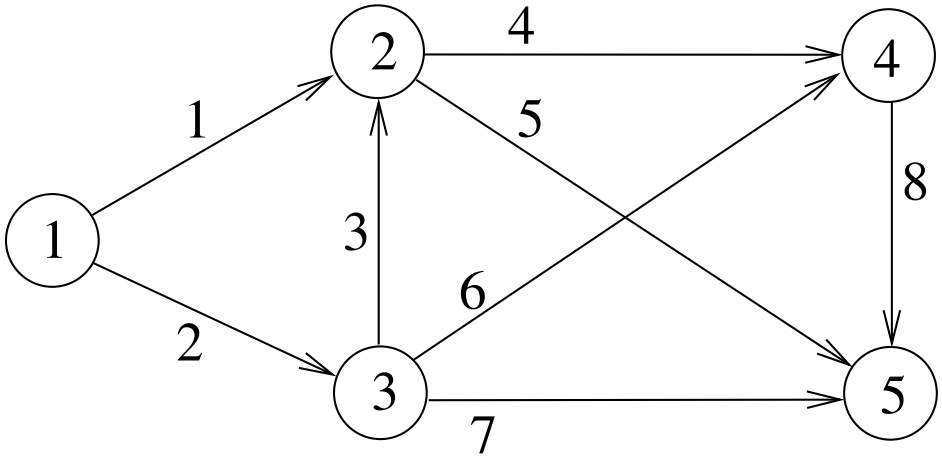
$$b_i = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases}$$

$$a_{ik} = \begin{cases} -1 & \text{om båge } k \text{ startar i nod } i \\ 1 & \text{om båge } k \text{ slutar i nod } i \\ 0 & \text{för övrigt} \end{cases}$$

A kallas anslutningsmatris.

En kolumn i A har ett element lika med -1, ett lika med 1 och resten nollor. Detta gäller även b.

Exempel



$$
\begin{array}{rrrrrrrrrl}
-x_{12} & -x_{13} & & & & & & & = & -1 \quad (1)\\
x_{12} & & +x_{32} & -x_{24} & -x_{25} & & & & = & 0 \quad (2)\\
& x_{13} & -x_{32} & & & -x_{34} & -x_{35} & & = & 0 \quad (3)\\
& & & x_{24} & & +x_{34} & & -x_{45} & = & 0 \quad (4)\\
& & & & x_{25} & & +x_{35} & +x_{45} & = & 1 \quad (5)
\end{array}
$$

Anslutningsmatris: $A = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$

Billigaste väg: Matematisk modell i vektor/matrisform

Definition

En matris A är **fullständigt unimodulär** om varje underdeterminant har värdet 0, 1 eller -1.

Billigaste väg: Matematisk modell i vektor/matrisform

Definition

En matris A är **fullständigt unimodulär** om varje underdeterminant har värdet 0, 1 eller -1.

Sats

Varje anslutningsmatris, A, är fullständigt unimodulär.

Billigaste väg: Matematisk modell i vektor/matrisform

Definition

En matris A är **fullständigt unimodulär** om varje underdeterminant har värdet 0, 1 eller -1.

Sats

Varje anslutningsmatris, A, är fullständigt unimodulär.

Sats

Om A är en fullständigt unimodulär matris och b är en heltalsvektor, så är alla extrempunkter till mängden $X = \{x : Ax = b,\ x \geq 0\}$ heltaliga.

Billigaste väg: Matematisk modell i vektor/matrisform

Definition

En matris A är **fullständigt unimodulär** om varje underdeterminant har värdet 0, 1 eller -1.

Sats

Varje anslutningsmatris, A, är fullständigt unimodulär.

Sats

Om A är en fullständigt unimodulär matris och b är en heltalsvektor, så är alla extrempunkter till mängden $X = \{x : Ax = b,\ x \geq 0\}$ heltaliga.

Sats

Ett LP-problem vars bivillkorsmatris är en anslutningsmatris och vars högerled är heltaligt, har heltalig optimallösning.

Billigaste väg: Matematisk modell i vektor/matrisform

Definition

En matris A är **fullständigt unimodulär** om varje underdeterminant har värdet 0, 1 eller -1.

Sats

Varje anslutningsmatris, A, är fullständigt unimodulär.

Sats

Om A är en fullständigt unimodulär matris och b är en heltalsvektor, så är alla extrempunkter till mängden $X = \{x : Ax = b,\ x \geq 0\}$ heltaliga.

Sats

Ett LP-problem vars bivillkorsmatris är en anslutningsmatris och vars högerled är heltaligt, har heltalig optimallösning.

Slutsats

Billigaste väg-problem kan betraktas som LP-problem.

Billigaste väg

LP-formulering:

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

$$\text{då} \quad \sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases} \quad \text{för alla } i \in N$$

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

Billigaste väg

LP-formulering:

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

$$\text{då} \quad \sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = \begin{cases} -1 & \text{då } i = s \\ 1 & \text{då } i = t \\ 0 & \text{f ö} \end{cases} \quad \text{för alla } i \in N$$

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

Slutsats

Om LP-formuleringen av billigaste väg-problemet har ändligt optimum, så är alla x_{ij} lika med 1 eller 0 i optimallösningen.

Lösningsmetoder för billigaste vägproblemet

Summering av $Ax = b$ ger

Lösningsmetoder för billigaste vägproblemet

Summering av $Ax = b$ ger $0 = 0$.

Lösningsmetoder för billigaste vägproblemet

Summering av $Ax = b$ ger $0 = 0$.

$Ax = b$ innehåller ett redundant bivillkor (raderna är linjärt beroende).

Lösningsmetoder för billigaste vägproblemet

Summering av $Ax = b$ ger $0 = 0$.

$Ax = b$ innehåller ett redundant bivillkor (raderna är linjärt beroende).

y_i: dualvariabel för nod i.

Lösningsmetoder för billigaste vägproblemet

Summering av $Ax = b$ ger $0 = 0$.

$Ax = b$ innehåller ett redundant bivillkor (raderna är linjärt beroende).

y_i: dualvariabel för nod i.

LP-dualen:

$$\begin{aligned} \max \quad & y_t - y_s \\ \text{då} \quad & y_j - y_i \leq c_{ij} \quad \text{för alla } (i,j) \end{aligned}$$

Lösningsmetoder för billigaste vägproblemet

Summering av $Ax = b$ ger $0 = 0$.

$Ax = b$ innehåller ett redundant bivillkor (raderna är linjärt beroende).

y_i: dualvariabel för nod i.

LP-dualen:

$$\begin{aligned} \max \quad & y_t - y_s \\ \text{då} \quad & y_j - y_i \leq c_{ij} \quad \text{för alla } (i,j) \end{aligned}$$

Ett redundant bivillkor i primalen ger en frihetsgrad i dualen. Sätt $y_s = 0$.

Lösningsmetoder för billigaste vägproblemet

Summering av $Ax = b$ ger $0 = 0$.

$Ax = b$ innehåller ett redundant bivillkor (raderna är linjärt beroende).

y_i: dualvariabel för nod i.

LP-dualen:

$$\begin{array}{ll} \max & y_t - y_s \\ \text{då} & y_j - y_i \leq c_{ij} \quad \text{för alla } (i,j) \end{array}$$

Ett redundant bivillkor i primalen ger en frihetsgrad i dualen. Sätt $y_s = 0$.

LP-dualitet ger:

Om y är en optimal duallösning (med $y_s = 0$), så är y_k är lägsta kostnaden för att komma till nod k från s.

Lösningsmetoder för billigaste vägproblemet

Summering av $Ax = b$ ger $0 = 0$.

$Ax = b$ innehåller ett redundant bivillkor (raderna är linjärt beroende).

y_i: dualvariabel för nod i.

LP-dualen:

$$\begin{array}{ll} \max & y_t - y_s \\ \text{då} & y_j - y_i \leq c_{ij} \quad \text{för alla } (i,j) \end{array}$$

Ett redundant bivillkor i primalen ger en frihetsgrad i dualen. Sätt $y_s = 0$.

LP-dualitet ger:

Om y är en optimal duallösning (med $y_s = 0$), så är y_k är lägsta kostnaden för att komma till nod k från s.

Vi kallar y_i nodpris.

Nodmärkningsmetoder

LP-dual: max y_t då $y_j - y_i \leq c_{ij}$ för alla (i, j) (samt $y_s = 0$)

Nodmärkningsmetoder

LP-dual: $\max\ y_t$ då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: $\max y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Nodmärkningsmetoder

LP-dual: $\max\ y_t$ då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: $\max y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Slutsats: Sätt $\boxed{y_j = \min_i(c_{ij} + y_i).}$

Nodmärkningsmetoder

LP-dual: max y_t då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: max $y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Slutsats: Sätt $\boxed{y_j = \min_i (c_{ij} + y_i).}$

Primal tolkning: Vi vill finna billigaste sättet att komma till nod j.

Nodmärkningsmetoder

LP-dual: max y_t då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: max $y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Slutsats: Sätt $\boxed{y_j = \min_i(c_{ij} + y_i).}$

Primal tolkning: Vi vill finna billigaste sättet att komma till nod j.

Märk nod j med (y_j, p_j), där p_j är det i som gav min (föregångaren).

Nodmärkningsmetoder

LP-dual: max y_t då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: max $y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Slutsats: Sätt $\boxed{y_j = \min_i(c_{ij} + y_i).}$

Primal tolkning: Vi vill finna billigaste sättet att komma till nod j.

Märk nod j med (y_j, p_j), där p_j är det i som gav min (föregångaren).

Praktisk fråga: I vilken ordning skall vi undersöka noderna?

Nodmärkningsmetoder

LP-dual: max y_t då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: max $y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Slutsats: Sätt $\boxed{y_j = \min_i(c_{ij} + y_i).}$

Primal tolkning: Vi vill finna billigaste sättet att komma till nod j.

Märk nod j med (y_j, p_j), där p_j är det i som gav min (föregångaren).

Praktisk fråga: I vilken ordning skall vi undersöka noderna?

Tre olika fall:

Nodmärkningsmetoder

LP-dual: max y_t då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: max $y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Slutsats: Sätt $\boxed{y_j = \min_i(c_{ij} + y_i).}$

Primal tolkning: Vi vill finna billigaste sättet att komma till nod j.

Märk nod j med (y_j, p_j), där p_j är det i som gav min (föregångaren).

Praktisk fråga: I vilken ordning skall vi undersöka noderna?

Tre olika fall:
1. Allmänt (negativa kostnader och cykler).

Nodmärkningsmetoder

LP-dual: max y_t då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: max $y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Slutsats: Sätt $\boxed{y_j = \min_i(c_{ij} + y_i).}$

Primal tolkning: Vi vill finna billigaste sättet att komma till nod j.

Märk nod j med (y_j, p_j), där p_j är det i som gav min (föregångaren).

Praktisk fråga: I vilken ordning skall vi undersöka noderna?

Tre olika fall:
1. Allmänt (negativa kostnader och cykler).
2. Inga negativa kostnader.

Nodmärkningsmetoder

LP-dual: $\max\ y_t$ då $y_j - y_i \leq c_{ij}$ för alla (i,j) (samt $y_s = 0$)

Bivillkoren: $y_j \leq c_{ij} + y_i$.

Målfunktionen: $\max y_t \Rightarrow$ öka alla nodpriser så mycket som möjligt.

Slutsats: Sätt $\boxed{y_j = \min_i(c_{ij} + y_i).}$

Primal tolkning: Vi vill finna billigaste sättet att komma till nod j.

Märk nod j med (y_j, p_j), där p_j är det i som gav min (föregångaren).

Praktisk fråga: I vilken ordning skall vi undersöka noderna?

Tre olika fall:
1. Allmänt (negativa kostnader och cykler).
2. Inga negativa kostnader.
3. Inga cykler.

Billigaste väg

Negativa kostnader och cykler: Fords metod

Billigaste väg

Negativa kostnader och cykler: Fords metod

Finns ingen ordning så att nodmärkningar säkert inte behöver göras om.

Billigaste väg

Negativa kostnader och cykler: Fords metod

Finns ingen ordning så att nodmärkningar säkert inte behöver göras om.

Avsökning av nod: Kolla utgående bågar. Uppdatera alla nodpriser som blir lägre. Om inget nodpris kan sänkas är noden avsökt.

Billigaste väg

Negativa kostnader och cykler: Fords metod

Finns ingen ordning så att nodmärkningar säkert inte behöver göras om.

Avsökning av nod: Kolla utgående bågar. Uppdatera alla nodpriser som blir lägre. Om inget nodpris kan sänkas är noden avsökt.

Om en avsökt nod får lägre nodpris, blir den oavsökt.

Billigaste väg

Negativa kostnader och cykler: Fords metod

Finns ingen ordning så att nodmärkningar säkert inte behöver göras om.

Avsökning av nod: Kolla utgående bågar. Uppdatera alla nodpriser som blir lägre. Om inget nodpris kan sänkas är noden avsökt.

Om en avsökt nod får lägre nodpris, blir den oavsökt.

Stoppkriterium: Alla noder avsökta.

Billigaste väg

Negativa kostnader och cykler: Fords metod

Finns ingen ordning så att nodmärkningar säkert inte behöver göras om.

Avsökning av nod: Kolla utgående bågar. Uppdatera alla nodpriser som blir lägre. Om inget nodpris kan sänkas är noden avsökt.

Om en avsökt nod får lägre nodpris, blir den oavsökt.

Stoppkriterium: Alla noder avsökta.

Cykel med negativ kostnad ger obegränsad lösning. Nodpriserna i cykeln kommer att uppdateras ett oändligt antal gånger. Man kan sluta efter $|N|$ gånger.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.

Om en avsökt nod får lägre nodpris, blir den oavsökt.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.

Om en avsökt nod får lägre nodpris, blir den oavsökt.

3. Markera nod k som avsökt.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.

Om en avsökt nod får lägre nodpris, blir den oavsökt.

3. Markera nod k som avsökt.
4. Om all noder är avsökta: Stopp.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.

Om en avsökt nod får lägre nodpris, blir den oavsökt.

3. Markera nod k som avsökt.
4. Om all noder är avsökta: Stopp.
5. Gå till 1.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.
Om en avsökt nod får lägre nodpris, blir den oavsökt.
3. Markera nod k som avsökt.
4. Om all noder är avsökta: Stopp.
5. Gå till 1.

Billigaste vägen nystas upp bakifrån mha p_j.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.
Om en avsökt nod får lägre nodpris, blir den oavsökt.
3. Markera nod k som avsökt.
4. Om all noder är avsökta: Stopp.
5. Gå till 1.

Billigaste vägen nystas upp bakifrån mha p_j.

Alla nodpriser kan ändras.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.
Om en avsökt nod får lägre nodpris, blir den oavsökt.
3. Markera nod k som avsökt.
4. Om all noder är avsökta: Stopp.
5. Gå till 1.

Billigaste vägen nystas upp bakifrån mha p_j.

Alla nodpriser kan ändras.

I steg 1 och 2 beaktas alla noder: Komplexitet $O(|N|^3)$.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.
Om en avsökt nod får lägre nodpris, blir den oavsökt.
3. Markera nod k som avsökt.
4. Om all noder är avsökta: Stopp.
5. Gå till 1.

Billigaste vägen nystas upp bakifrån mha p_j.

Alla nodpriser kan ändras.

I steg 1 och 2 beaktas alla noder: Komplexitet $O(|N|^3)$.

Obs: Först görs nodmärkningen helt färdigt, dvs. den duala lösningen finnes.

Billigaste väg: Fords metod

0. Sätt $y_s = 0$ och $y_j = M$ för alla andra noder.
1. Finn oavsökt nod (k) med lägsta nodpris $(\min_j y_j)$.
2. Uppdatera de nodpriser som blir lägre via y_k.
Om en avsökt nod får lägre nodpris, blir den oavsökt.
3. Markera nod k som avsökt.
4. Om all noder är avsökta: Stopp.
5. Gå till 1.

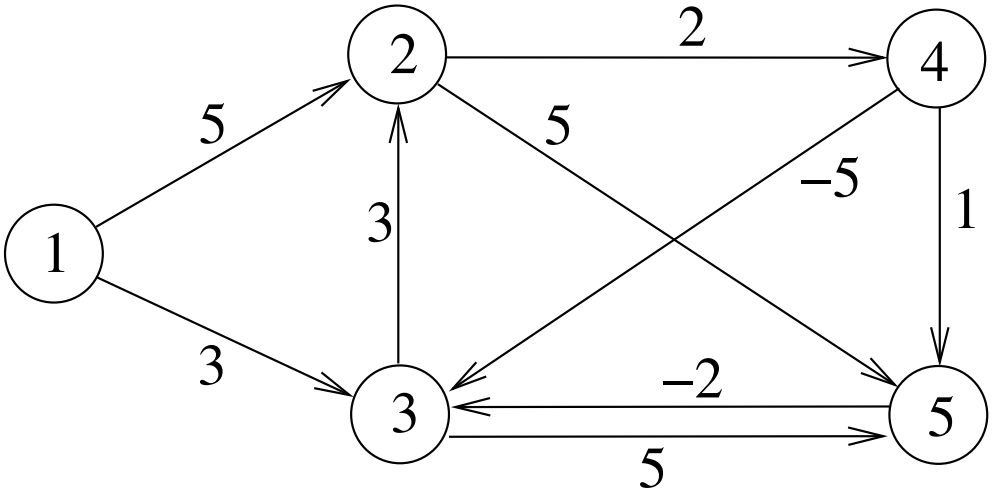
Billigaste vägen nystas upp bakifrån mha p_j.

Alla nodpriser kan ändras.

I steg 1 och 2 beaktas alla noder: Komplexitet $O(|N|^3)$.

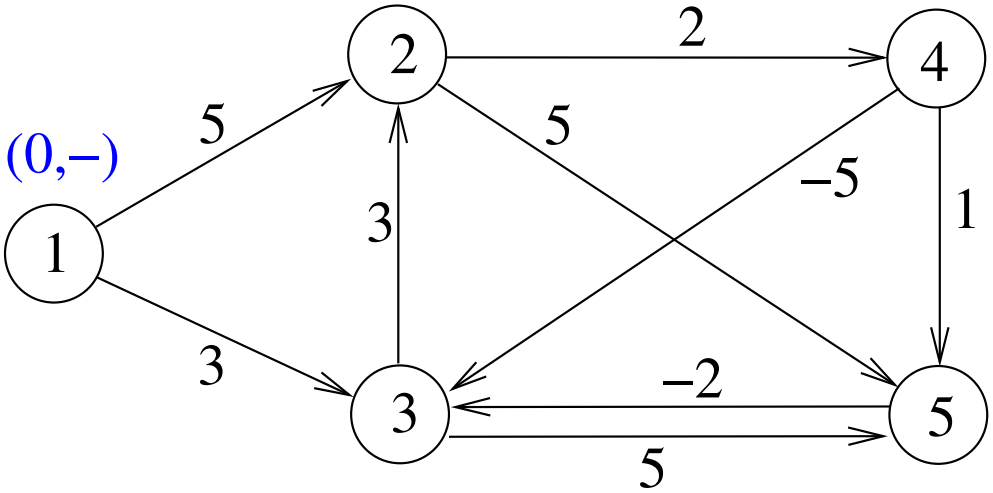
Obs: Först görs nodmärkningen helt färdigt, dvs. den duala lösningen finnes. Sedan hittar man vägen.

Fords metod för billigaste väg: Exempel



Lös med Fords metod.

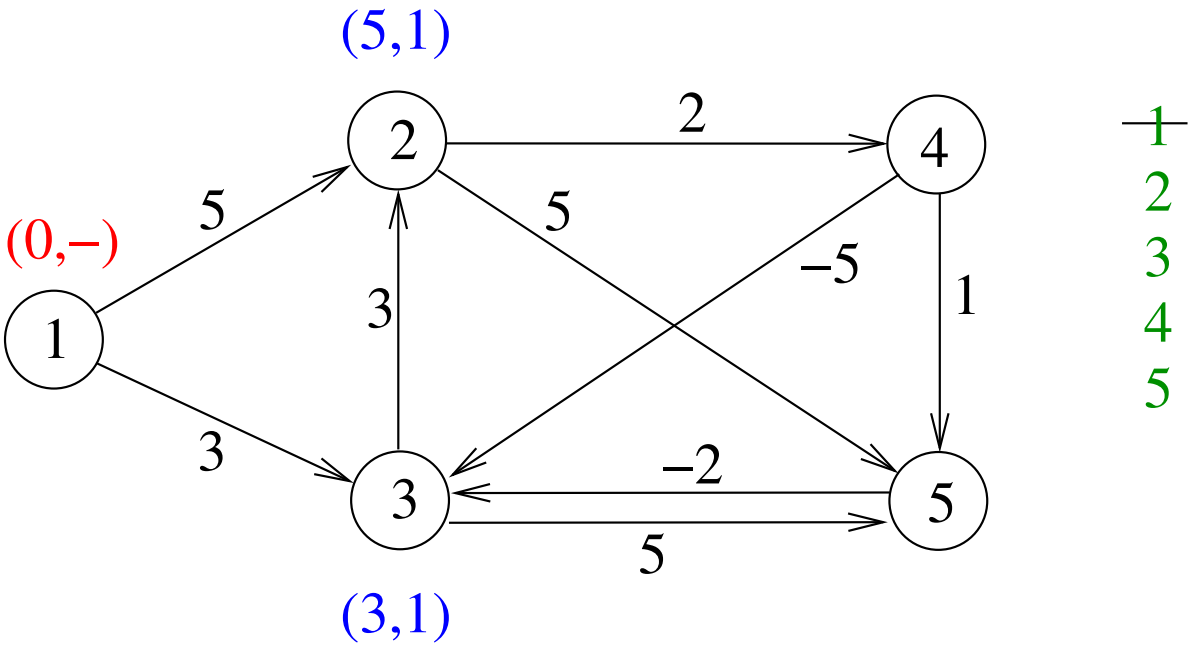
Fords metod för billigaste väg: Exempel



1
2
3
4
5

Märk nod 1.

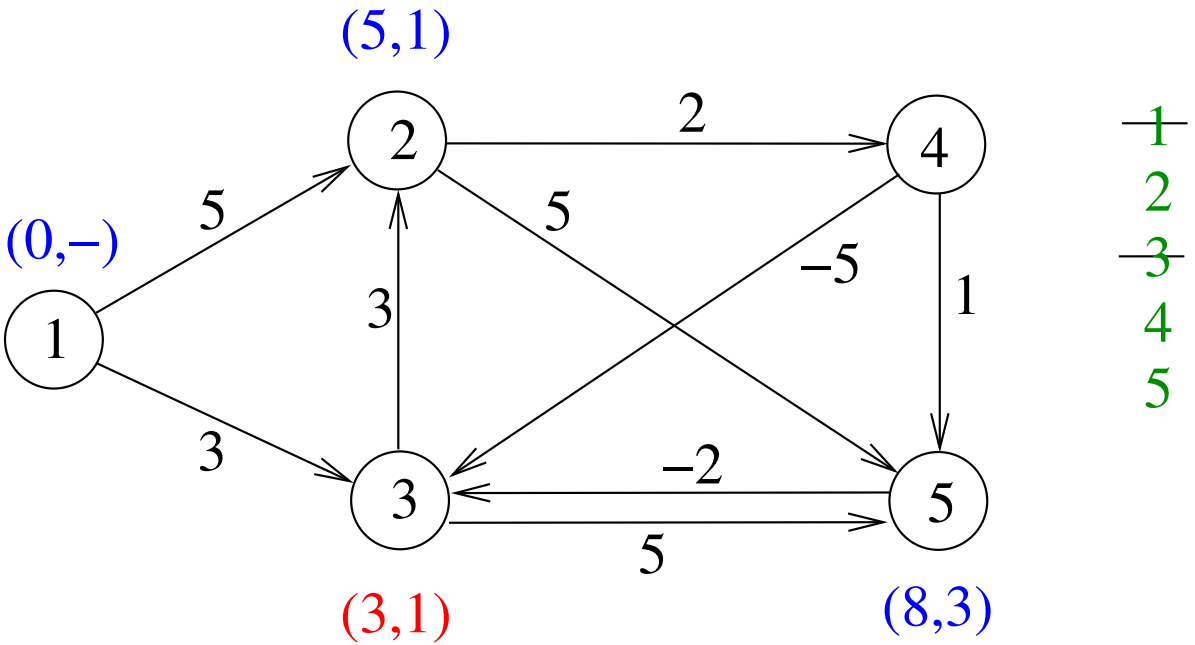
Fords metod för billigaste väg: Exempel



~~1~~
2
3
4
5

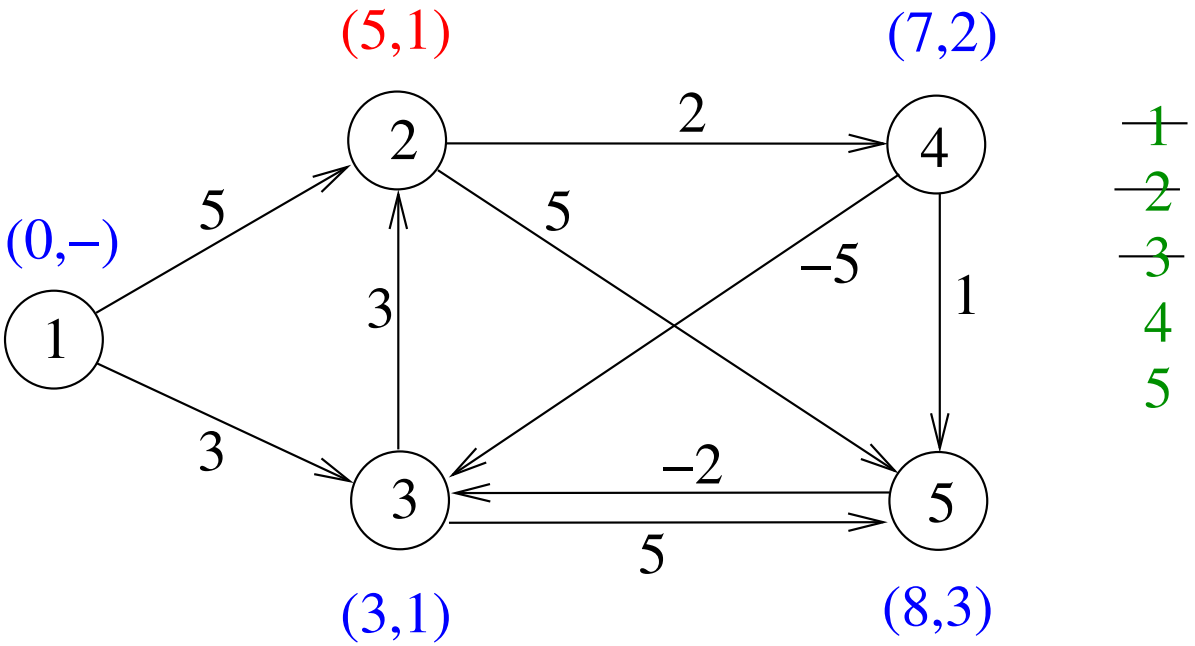
Märk nod 2 och 3 utifrån nod 1. Nod 1 avsökt.

Fords metod för billigaste väg: Exempel



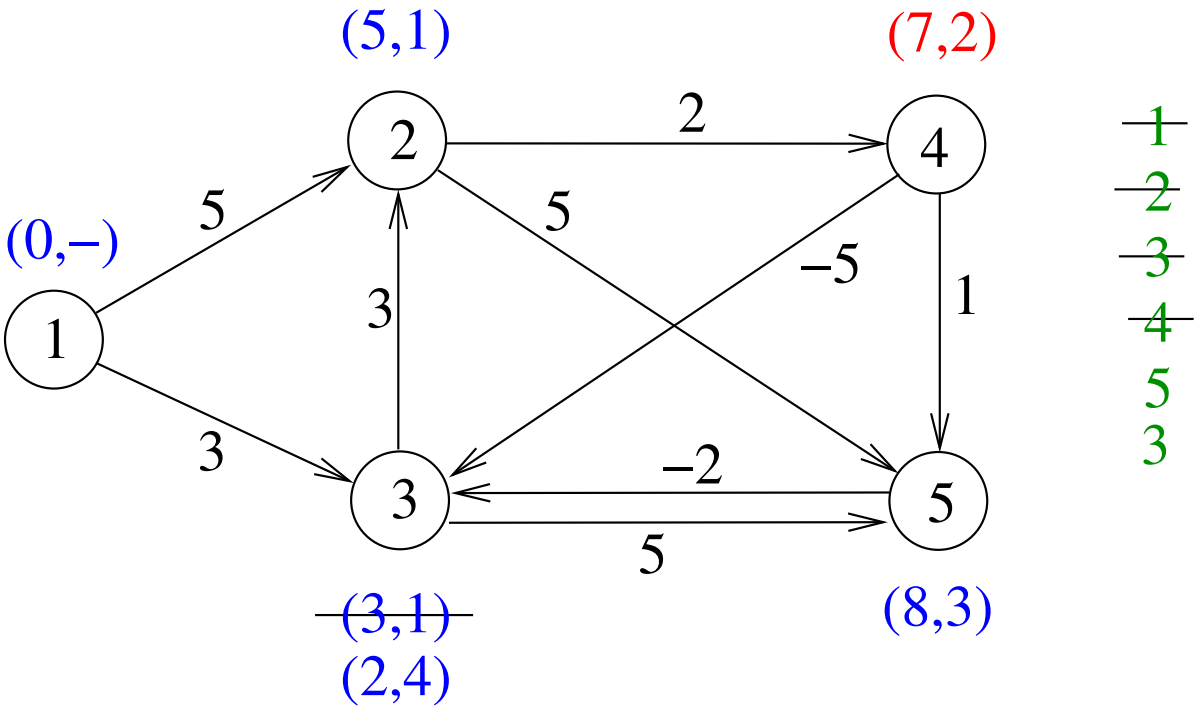
Märk/kolla nod 2 och 5 utifrån nod 3. Nod 3 avsökt.

Fords metod för billigaste väg: Exempel



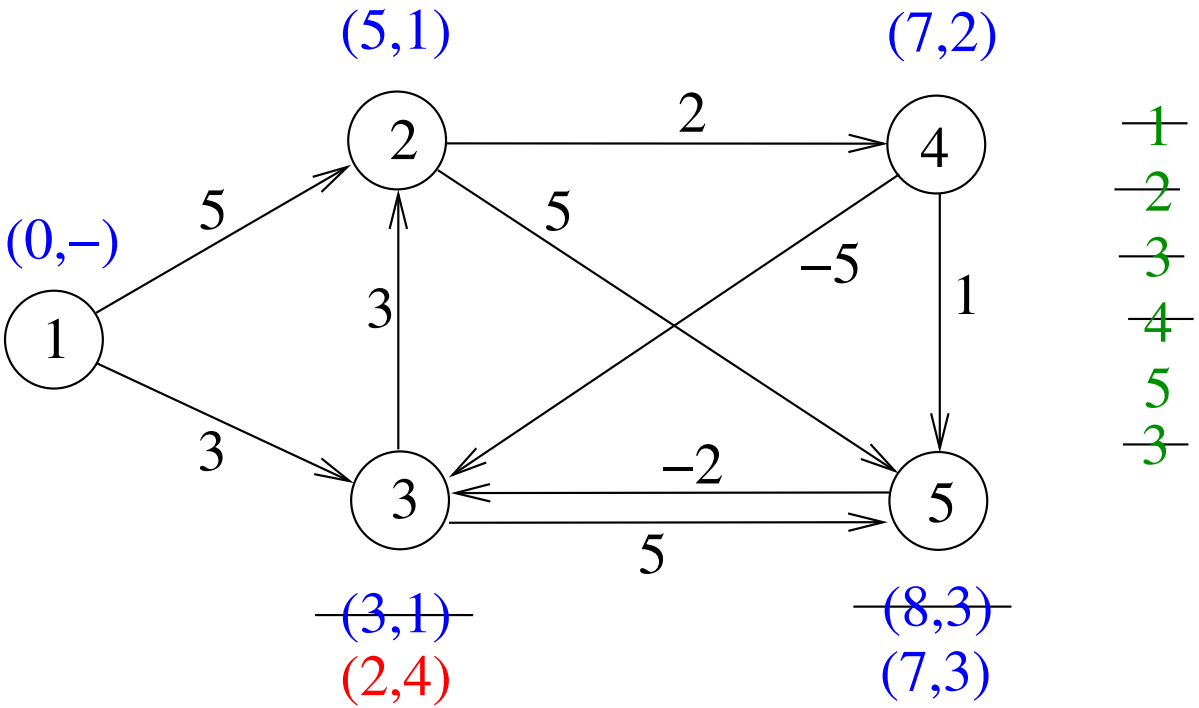
Märk/kolla nod 4 och 5 utifrån nod 2. Nod 2 avsökt.

Fords metod för billigaste väg: Exempel



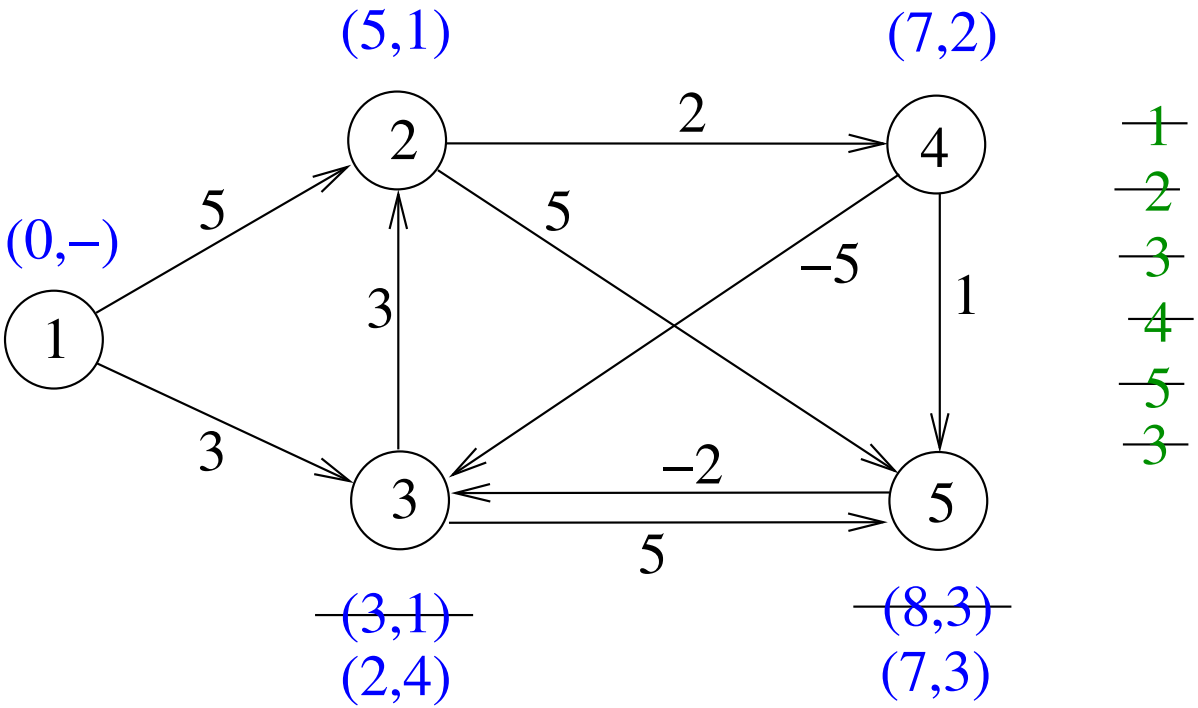
Märk/kolla nod 3 och 5 utifrån nod 4. Nod 4 avsökt. Nod 3 blir oavsökt.

Fords metod för billigaste väg: Exempel



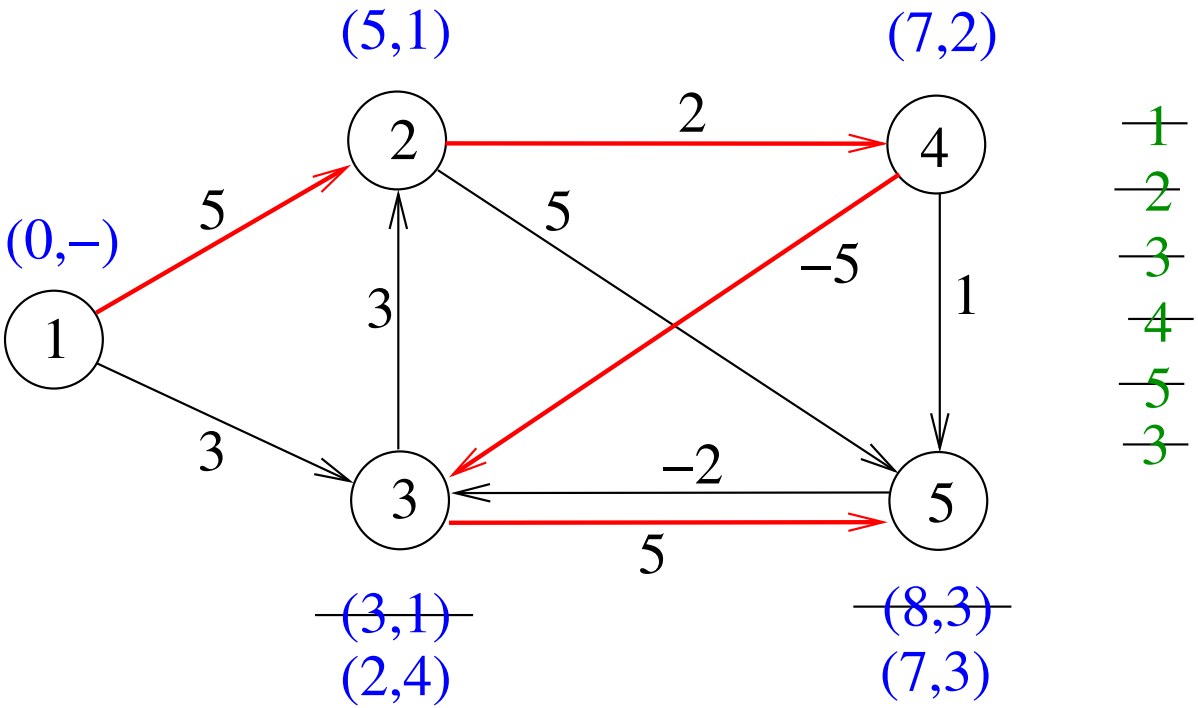
Märk/kolla nod 2 och 5 utifrån nod 3. Nod 3 avsökt.

Fords metod för billigaste väg: Exempel



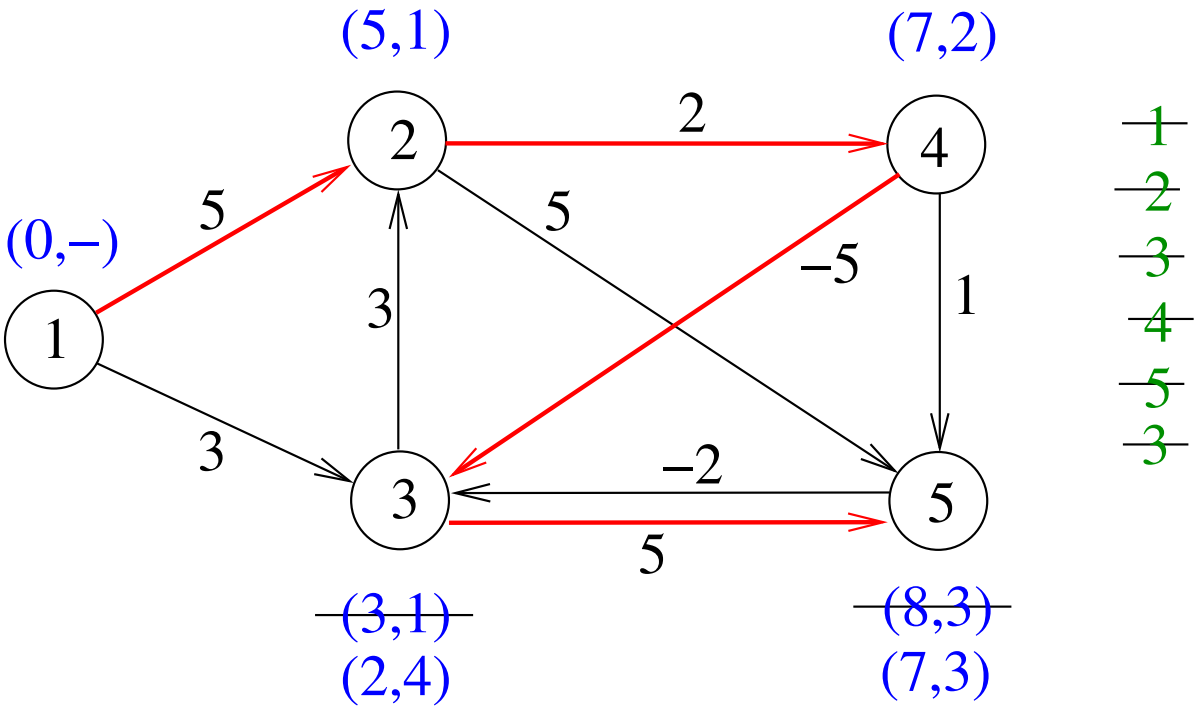
Märk/kolla nod 3 utifrån nod 5. Nod 5 avsökt.

Fords metod för billigaste väg: Exempel



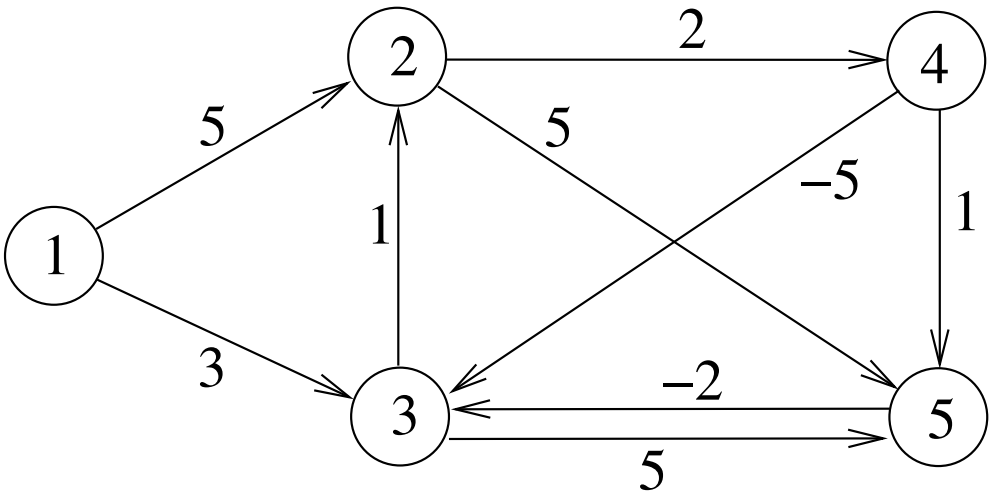
Alla noder avsökta. Finn väg baklänges.

Fords metod för billigaste väg: Exempel



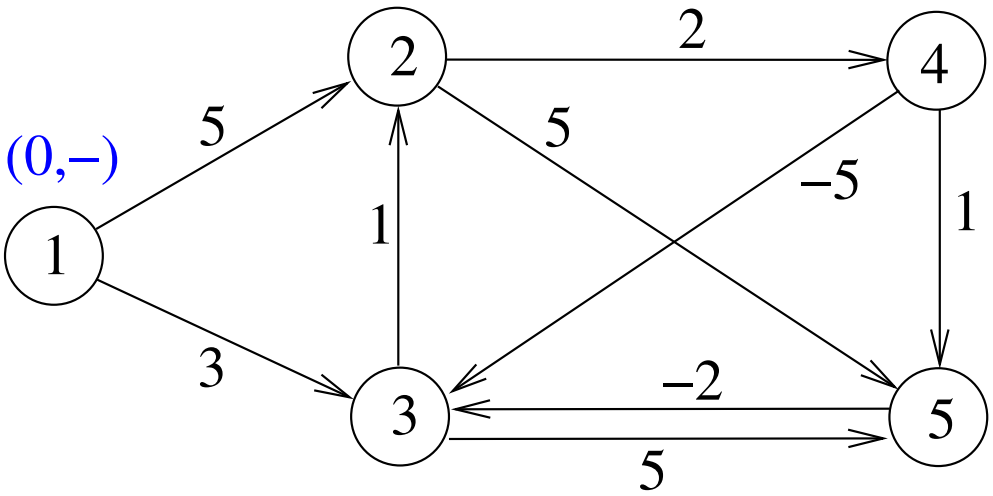
Alla noder avsökta. Finn väg baklänges.
Billigaste väg: 1 - 2 - 4 - 3 - 5. Kostnad: 7.

Fords metod för billigaste väg: Exempel 2



Lös med Fords metod.

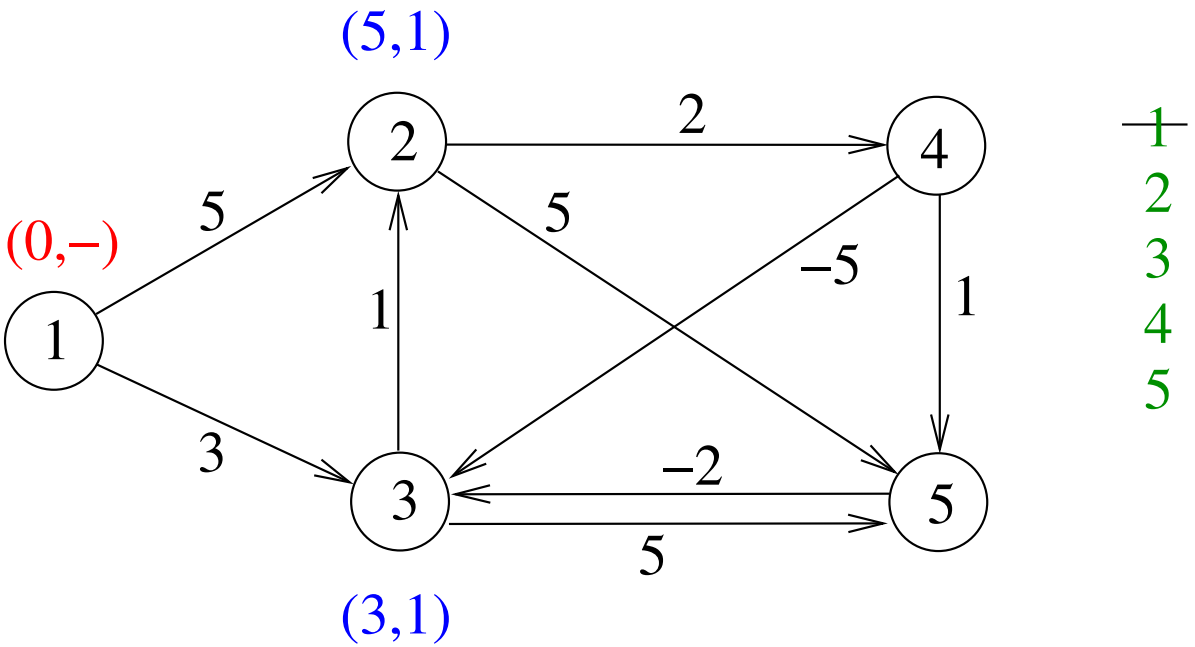
Fords metod för billigaste väg: Exempel 2



1
2
3
4
5

Märk nod 1.

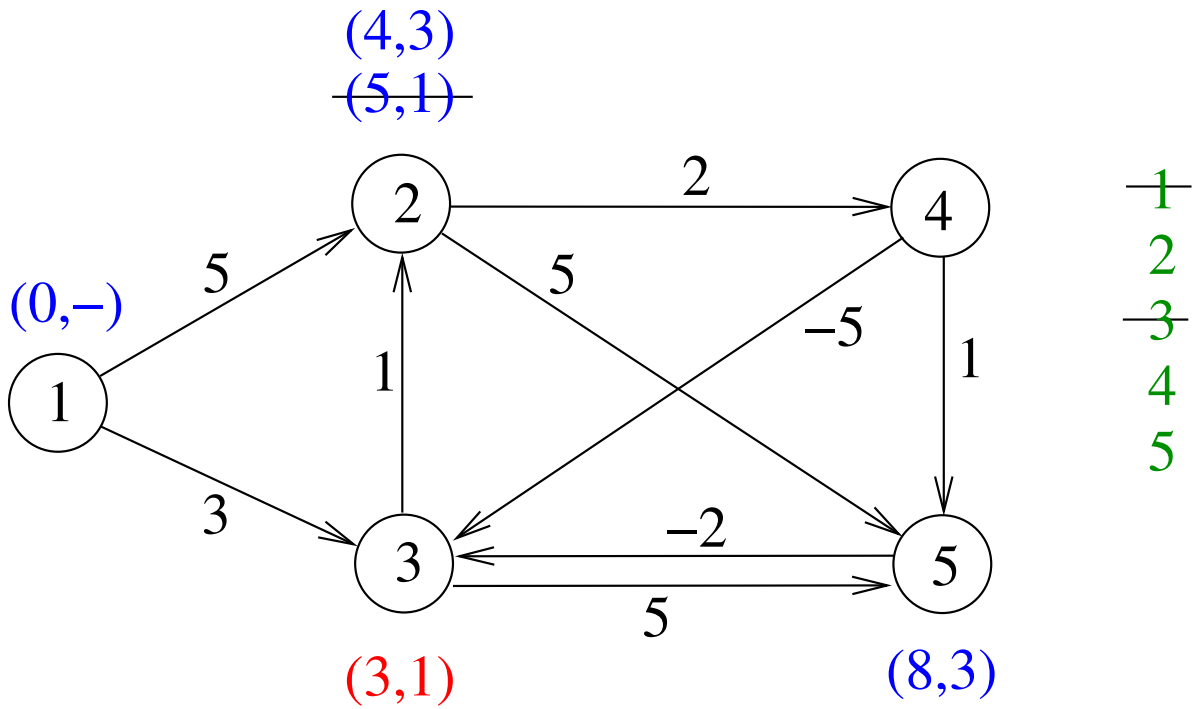
Fords metod för billigaste väg: Exempel 2



~~1~~
2
3
4
5

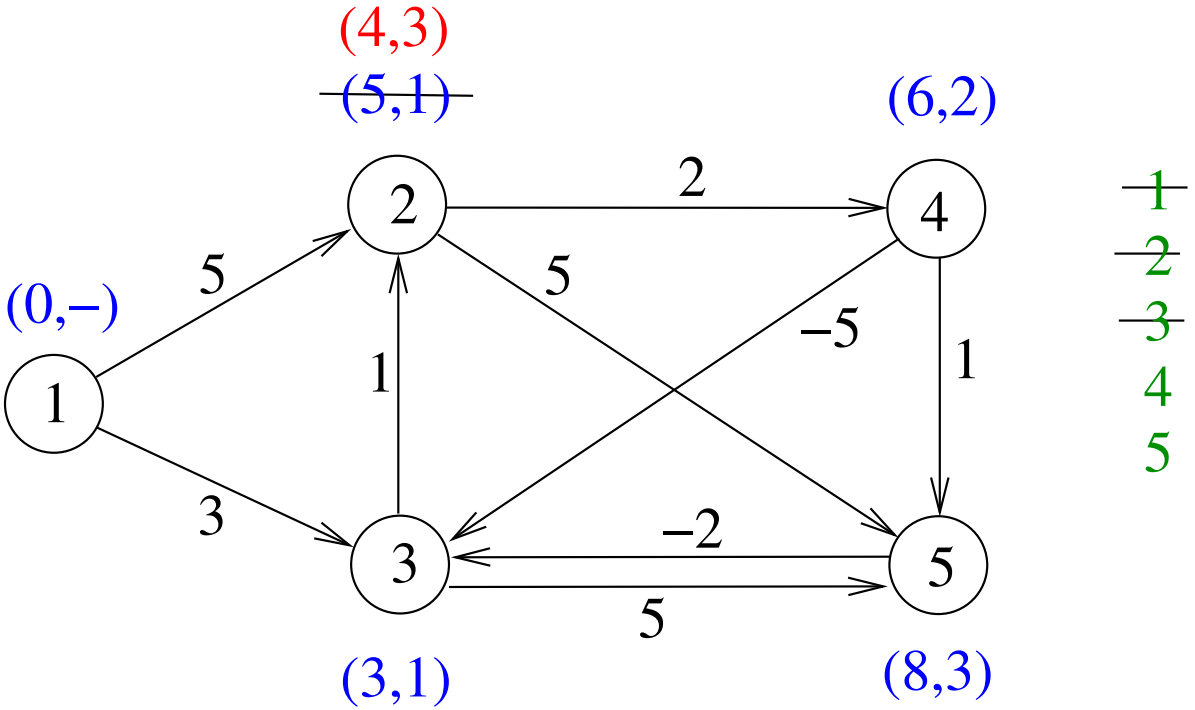
Märk nod 2 och 3 utifrån nod 1. Nod 1 avsökt.

Fords metod för billigaste väg: Exempel 2



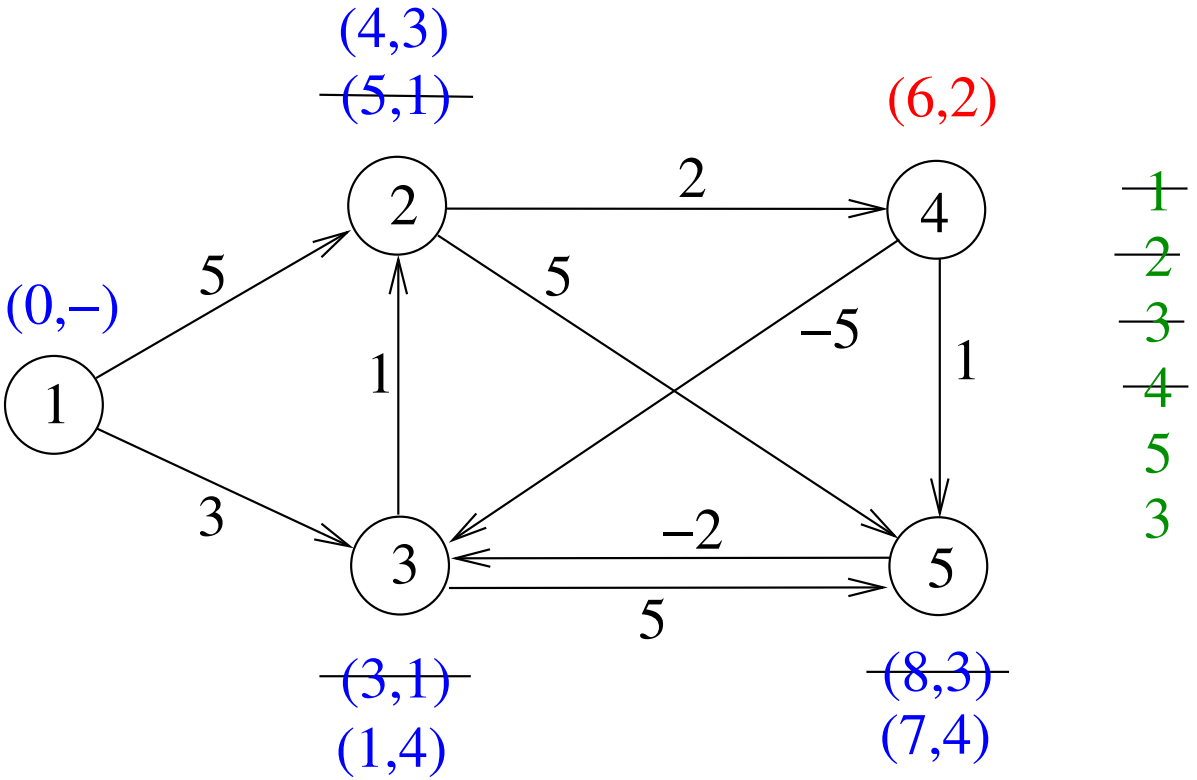
Märk/kolla nod 2 och 5 utifrån nod 3. Nod 3 avsökt.

Fords metod för billigaste väg: Exempel 2



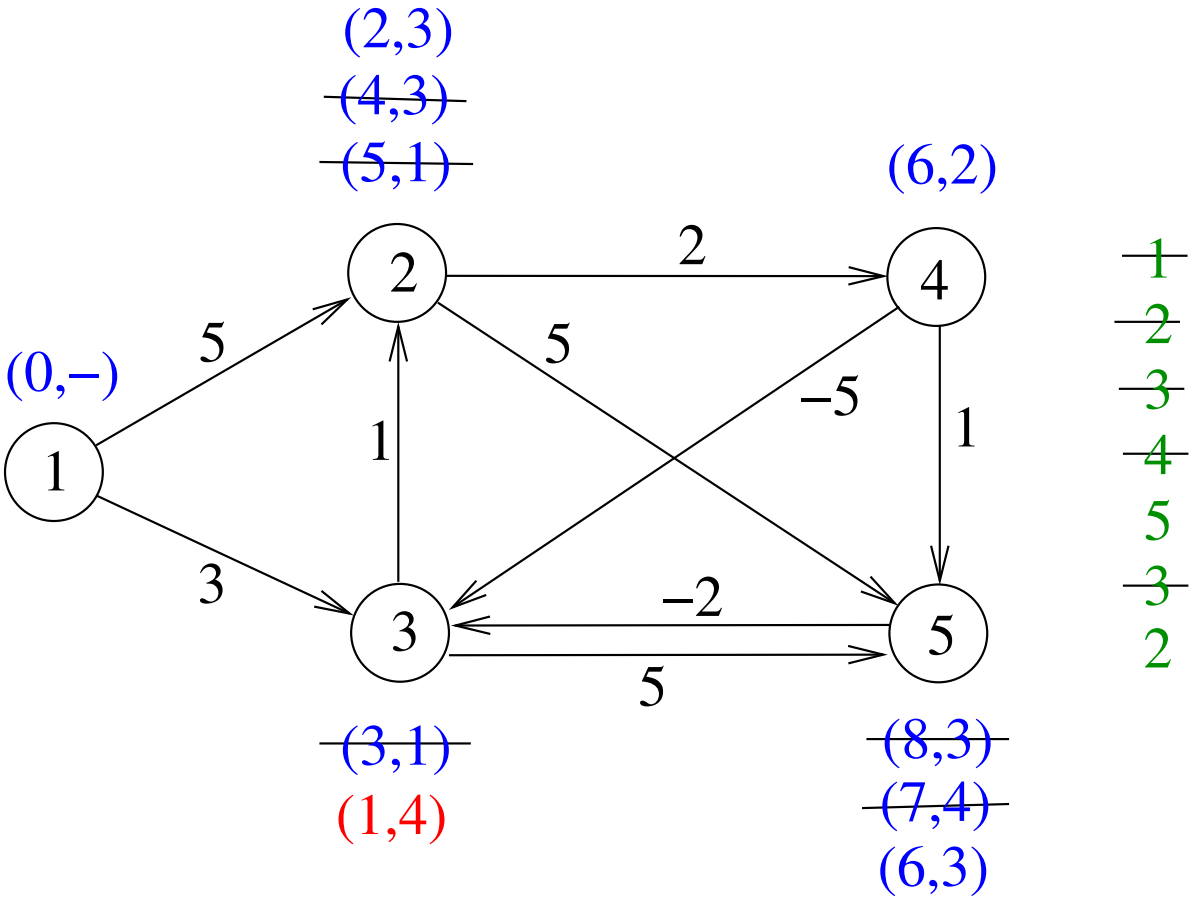
Märk/kolla nod 4 och 5 utifrån nod 2. Nod 2 avsökt.

Fords metod för billigaste väg: Exempel 2



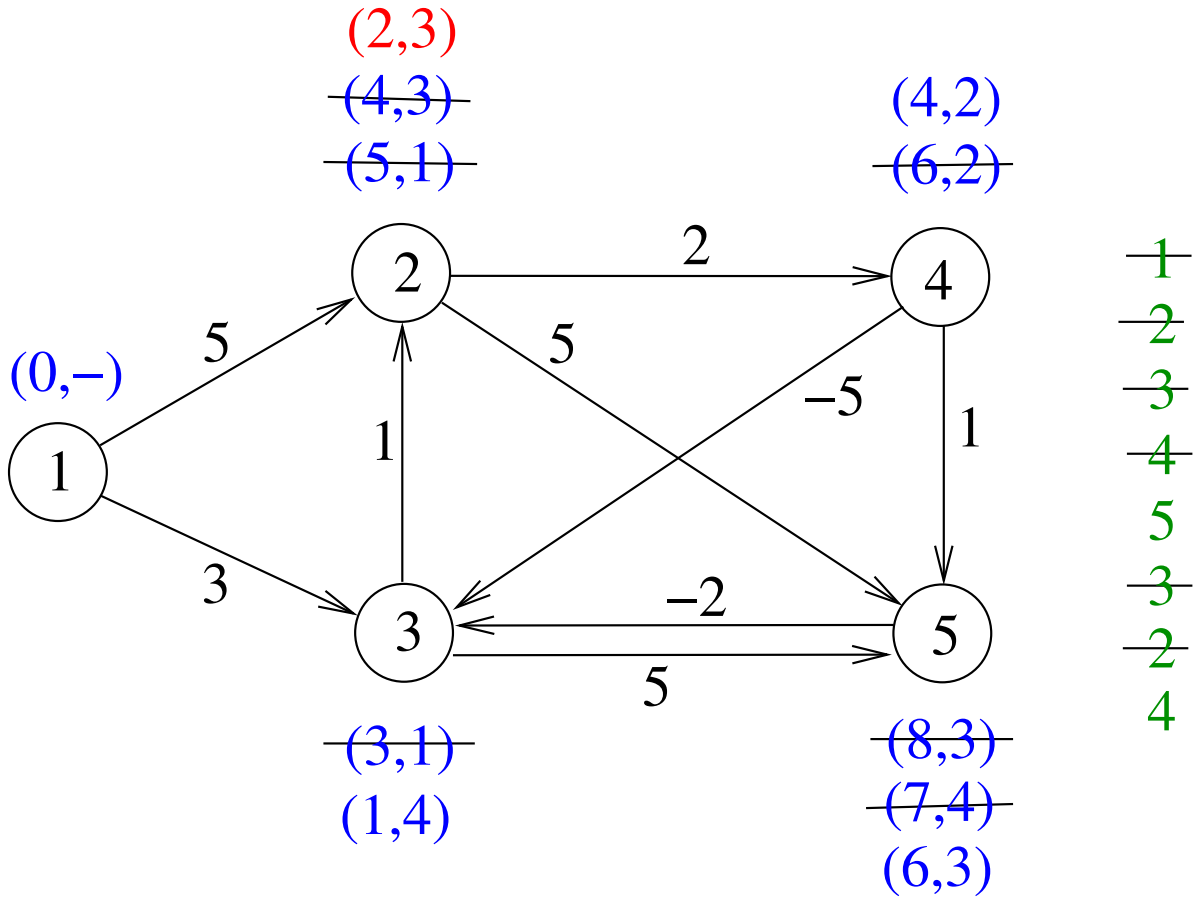
Märk/kolla nod 3 och 5 utifrån nod 4. Nod 4 avsökt. Nod 3 blir oavsökt.

Fords metod för billigaste väg: Exempel 2



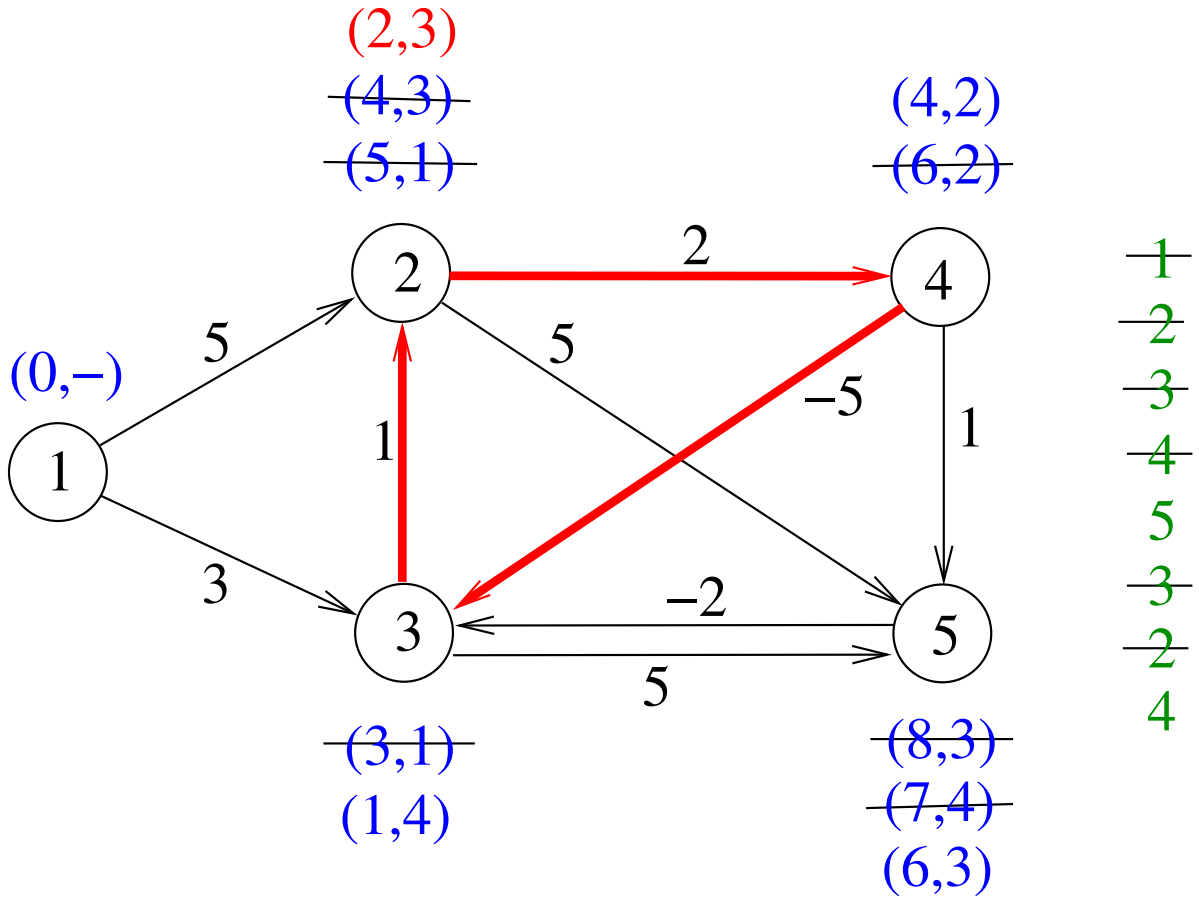
Märk/kolla nod 2 och 5 utifrån nod 3. Nod 3 avsökt. Nod 2 blir oavsökt.

Fords metod för billigaste väg: Exempel 2



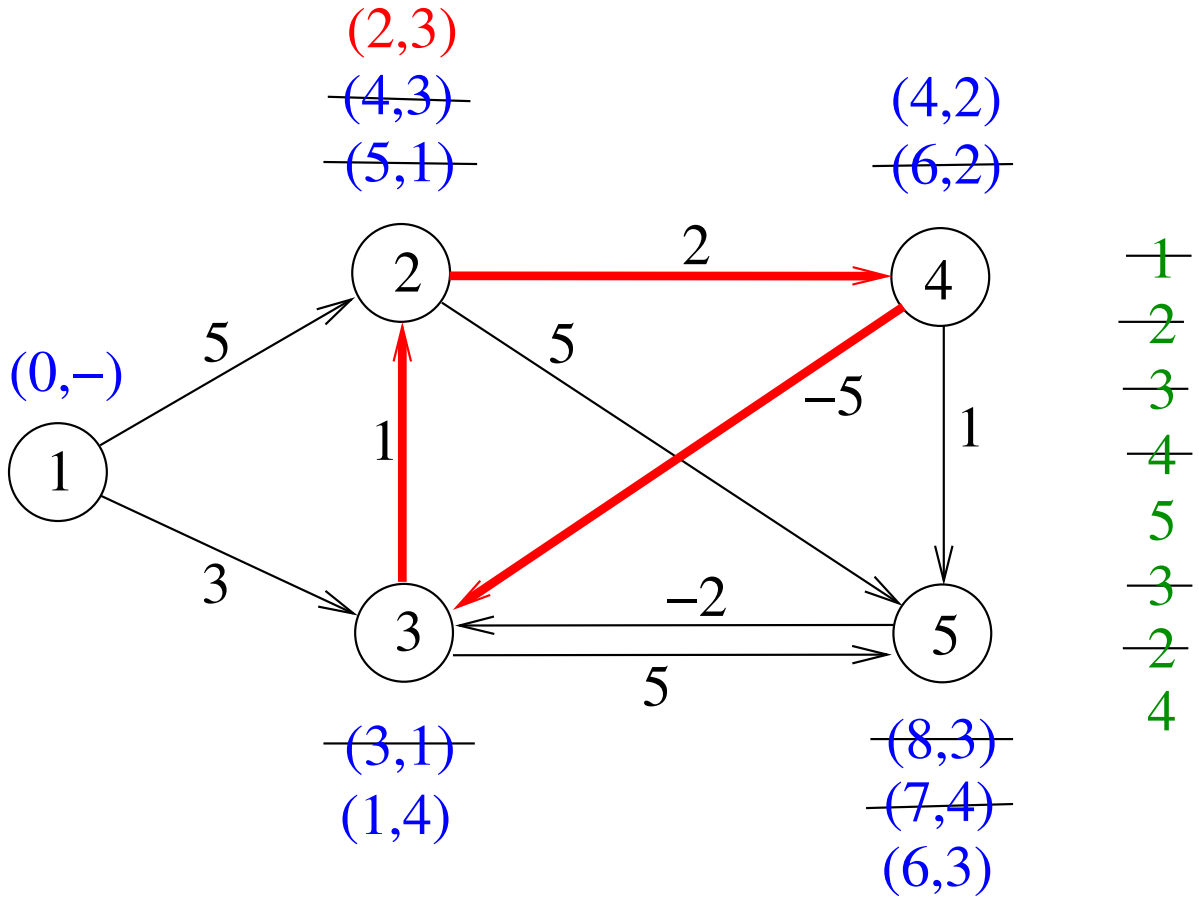
Märk/kolla nod 4 och 5 utifrån nod 2. Nod 2 avsökt. Nod 4 blir oavsökt.

Fords metod för billigaste väg: Exempel 2



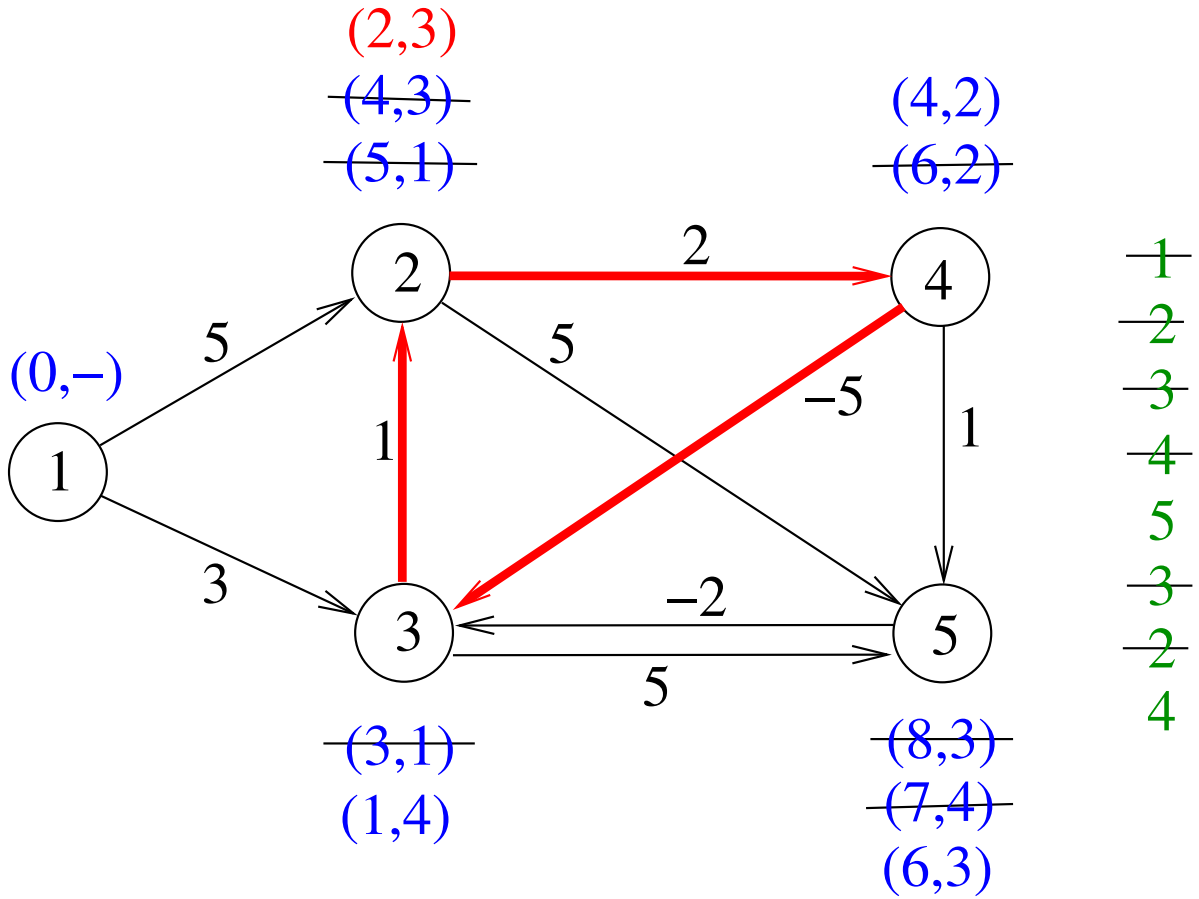
Vi kommer att fortsätta sänka nodpriserna för nod 4, 3, 2.

Fords metod för billigaste väg: Exempel 2



Vi kommer att fortsätta sänka nodpriserna för nod 4, 3, 2.
Negativ cykel: 2 - 4 - 3 - 2.

Fords metod för billigaste väg: Exempel 2

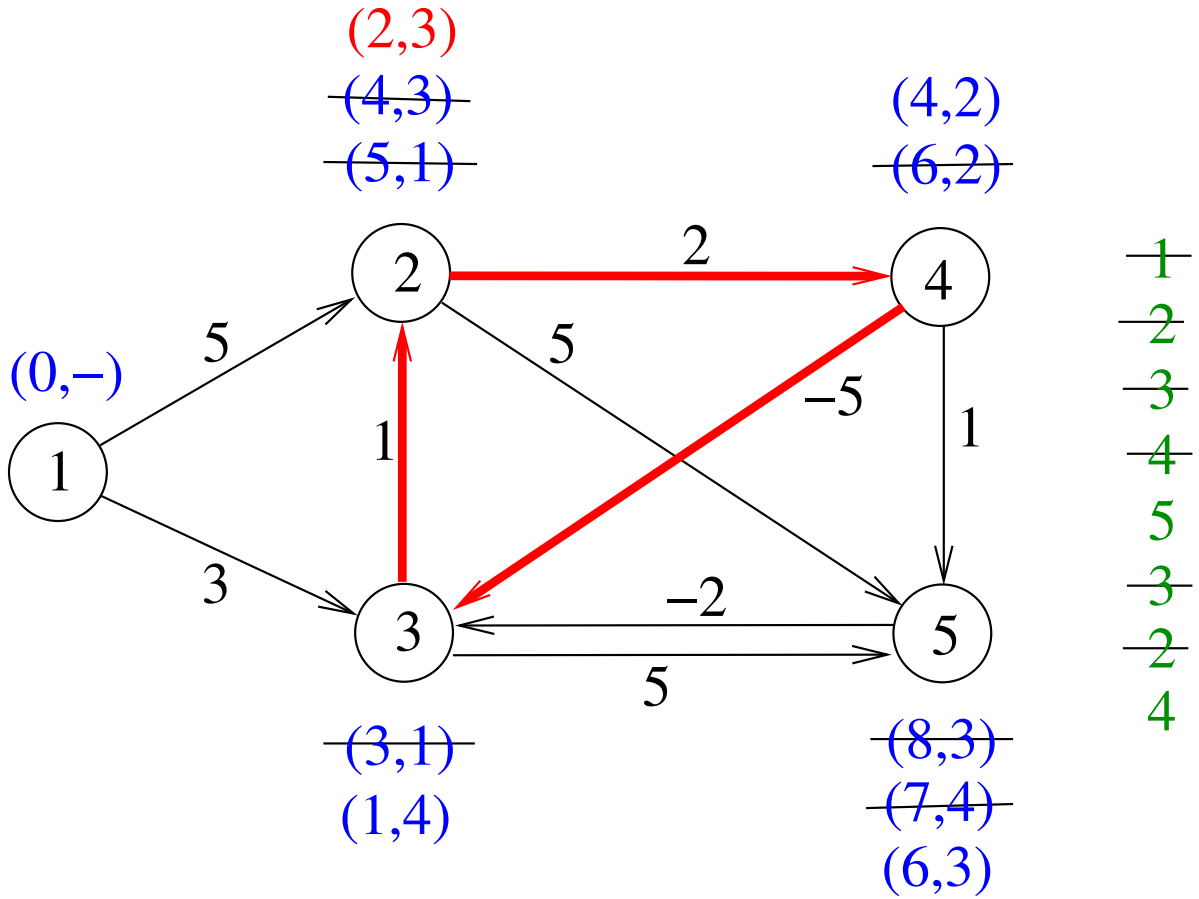


Vi kommer att fortsätta sänka nodpriserna för nod 4, 3, 2.

Negativ cykel: 2 - 4 - 3 - 2.

Oändligt bra primal lösning.

Fords metod för billigaste väg: Exempel 2



Vi kommer att fortsätta sänka nodpriserna för nod 4, 3, 2.
Negativ cykel: 2 - 4 - 3 - 2.
Oändligt bra primal lösning.
Ingen tillåten duallösning.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras,

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Det lägsta temporära nodpriset görs permanent.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Det lägsta temporära nodpriset görs permanent.

Frånvaron av negativa kostnader gör att inga nodpriser kan sänkas genom att gå i ytterligare en båge.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Det lägsta temporära nodpriset görs permanent.

Frånvaron av negativa kostnader gör att inga nodpriser kan sänkas genom att gå i ytterligare en båge.

Det blir alltid lite dyrare.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Det lägsta temporära nodpriset görs permanent.

Frånvaron av negativa kostnader gör att inga nodpriser kan sänkas genom att gå i ytterligare en båge.

Det blir alltid lite dyrare. Minimera fördyringen.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Det lägsta temporära nodpriset görs permanent.

Frånvaron av negativa kostnader gör att inga nodpriser kan sänkas genom att gå i ytterligare en båge.

Det blir alltid lite dyrare. Minimera fördyringen.

De permanenta nodmärkningar är minimala och behöver inte ändras.

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Det lägsta temporära nodpriset görs permanent.

Frånvaron av negativa kostnader gör att inga nodpriser kan sänkas genom att gå i ytterligare en båge.

Det blir alltid lite dyrare. Minimera fördyringen.

De permanenta nodmärkningar är minimala och behöver inte ändras.

(Försök inte ens.)

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Det lägsta temporära nodpriset görs permanent.

Frånvaron av negativa kostnader gör att inga nodpriser kan sänkas genom att gå i ytterligare en båge.

Det blir alltid lite dyrare. Minimera fördyringen.

De permanenta nodmärkningar är minimala och behöver inte ändras.

(Försök inte ens.)

Effektivare än Fords metod,

Nodmärkningsmetoder

Ickenegativa kostnader: Dijkstras metod

Arbeta med temporära och permanenta nodmärkningar.

De temporära ändras, och görs till sist permanenta.

De permanenta ändras inte.

Det lägsta temporära nodpriset görs permanent.

Frånvaron av negativa kostnader gör att inga nodpriser kan sänkas genom att gå i ytterligare en båge.

Det blir alltid lite dyrare. Minimera fördyringen.

De permanenta nodmärkningar är minimala och behöver inte ändras.

(Försök inte ens.)

Effektivare än Fords metod, men kanske lite krångligare.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

2. Om slutnoden (eller alla noder) permanent märkt: Stopp.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

2. Om slutnoden (eller alla noder) permanent märkt: Stopp.

3. Uppdatera de temporära nodpriser som blir lägre via y_k, dvs. om $c_{kj} + y_k < y_j$ för $j \notin A$.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

2. Om slutnoden (eller alla noder) permanent märkt: Stopp.

3. Uppdatera de temporära nodpriser som blir lägre via y_k, dvs. om $c_{kj} + y_k < y_j$ för $j \notin A$.

4. Gå till 1.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

2. Om slutnoden (eller alla noder) permanent märkt: Stopp.

3. Uppdatera de temporära nodpriser som blir lägre via y_k, dvs. om $c_{kj} + y_k < y_j$ för $j \notin A$.

4. Gå till 1.

Viktigt för effektiviteten: Titta inte på permanent märkta noder, $j \in A$, i steg 1 och 3.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

2. Om slutnoden (eller alla noder) permanent märkt: Stopp.

3. Uppdatera de temporära nodpriser som blir lägre via y_k, dvs. om $c_{kj} + y_k < y_j$ för $j \notin A$.

4. Gå till 1.

Viktigt för effektiviteten: Titta inte på permanent märkta noder, $j \in A$, i steg 1 och 3. Komplexitet $O(|N|^2)$.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

2. Om slutnoden (eller alla noder) permanent märkt: Stopp.

3. Uppdatera de temporära nodpriser som blir lägre via y_k, dvs. om $c_{kj} + y_k < y_j$ för $j \notin A$.

4. Gå till 1.

Viktigt för effektiviteten: Titta inte på permanent märkta noder, $j \in A$, i steg 1 och 3. Komplexitet $O(|N|^2)$.

Billigaste vägen nystas upp bakifrån med föregångare p_j (börja med $j = t$).

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

2. Om slutnoden (eller alla noder) permanent märkt: Stopp.

3. Uppdatera de temporära nodpriser som blir lägre via y_k, dvs. om $c_{kj} + y_k < y_j$ för $j \notin A$.

4. Gå till 1.

Viktigt för effektiviteten: Titta inte på permanent märkta noder, $j \in A$, i steg 1 och 3. Komplexitet $O(|N|^2)$.

Billigaste vägen nystas upp bakifrån med föregångare p_j (börja med $j = t$).

Obs: Först görs nodmärkningen helt färdigt.

Billigaste väg: Dijkstras metod

A: permanent märkta noder.

0. Sätt $y_s = 0$, $y_j = c_{sj}$ om båge (s, j) finns, och $y_j = M$ för alla andra $j \neq s$, samt $A = \{s\}$.

1. Finn nod (k) med lägsta temporära nodpris: $y_k = \min_{j \notin A} y_j$.

Lägg till k till A, dvs. gör märkningen permanent.

2. Om slutnoden (eller alla noder) permanent märkt: Stopp.

3. Uppdatera de temporära nodpriser som blir lägre via y_k, dvs. om $c_{kj} + y_k < y_j$ för $j \notin A$.

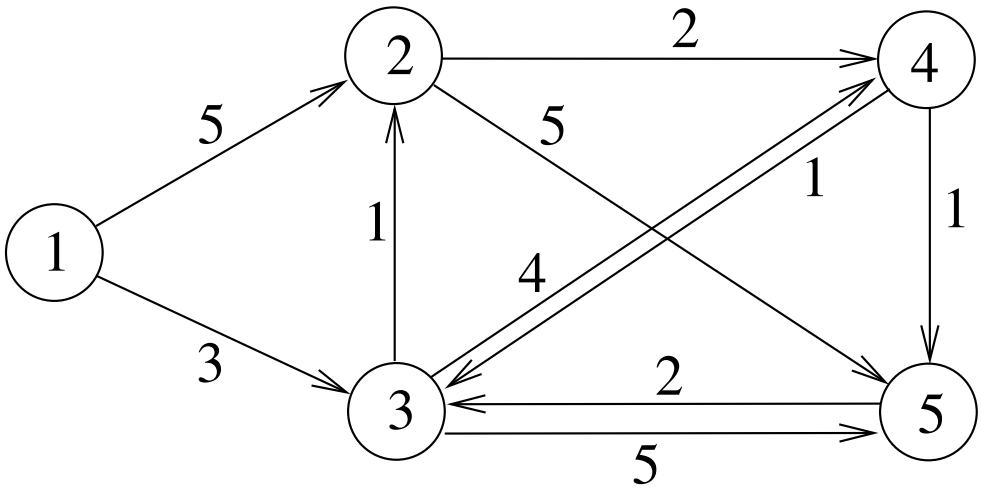
4. Gå till 1.

Viktigt för effektiviteten: Titta inte på permanent märkta noder, $j \in A$, i steg 1 och 3. Komplexitet $O(|N|^2)$.

Billigaste vägen nystas upp bakifrån med föregångare p_j (börja med $j = t$).

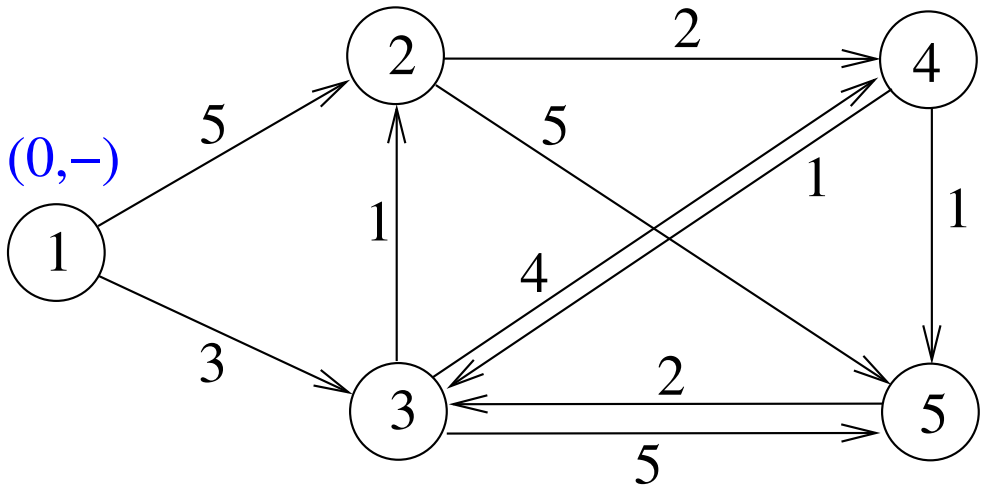
Obs: Först görs nodmärkningen helt färdigt. Sedan hittar man vägen.

Dijkstras metod för billigaste väg: Exempel



Lös med Dijkstras metod.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

2: (5,1)

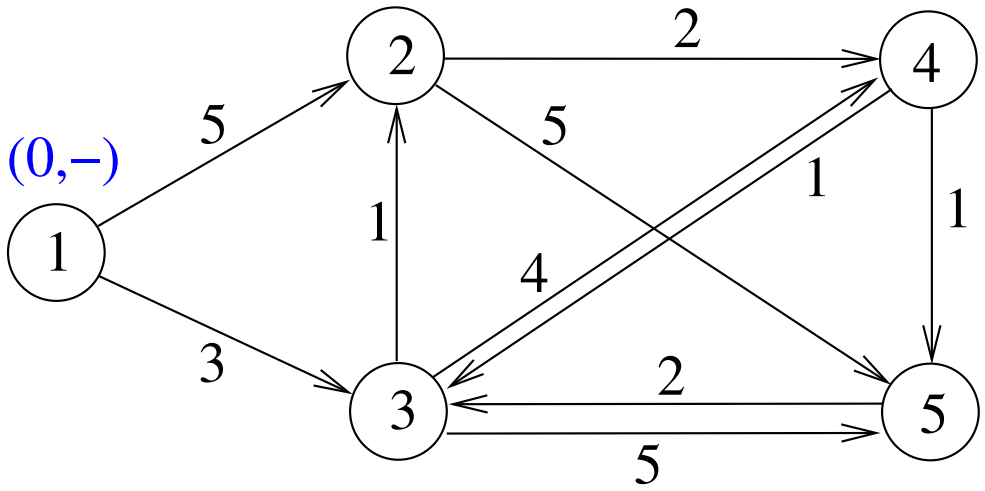
3: (3,1)

4: M

5: M

Märk nod 1 permanent, samt därefter 2 och 3 temporärt.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

2: (5,1)

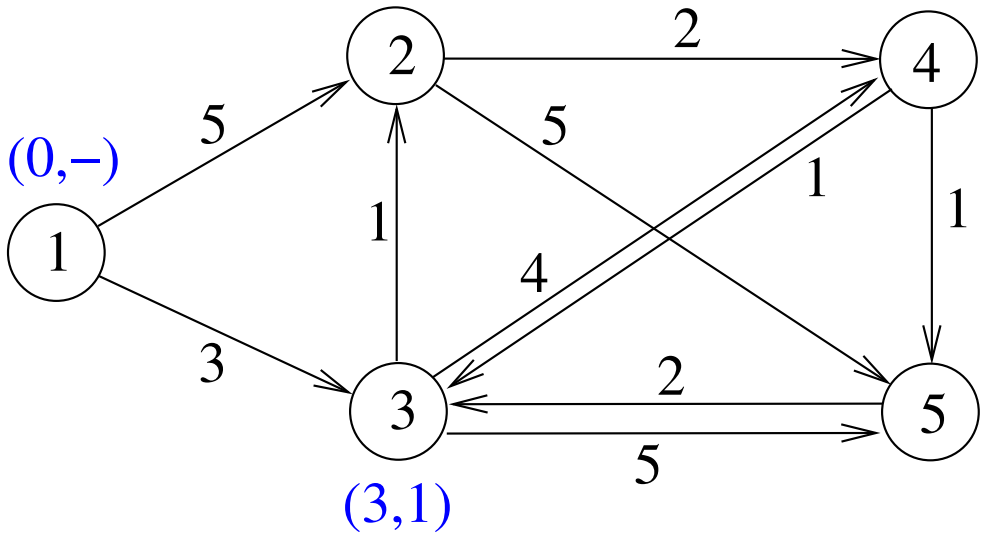
3: (3,1)

4: M

5: M

Nod 3 är bäst av de temporära.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

2: ~~(5,1)~~, (4,3)

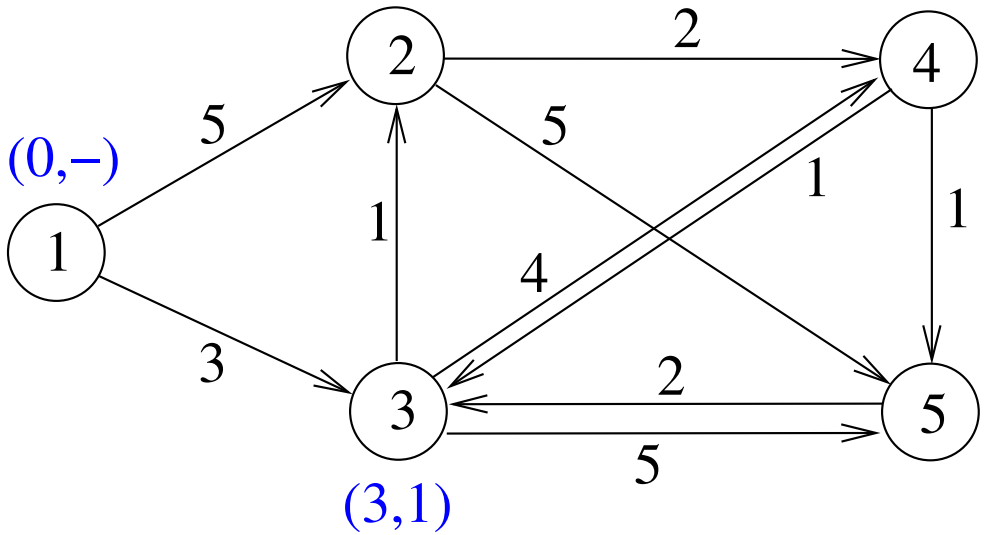
~~3: (3,1)~~

4: (7,3)

5: (8,3)

Märk nod 3 permanent. Nod 2, 4 och 5 får bättre temp.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

2: ~~(5,1)~~, (4,3)

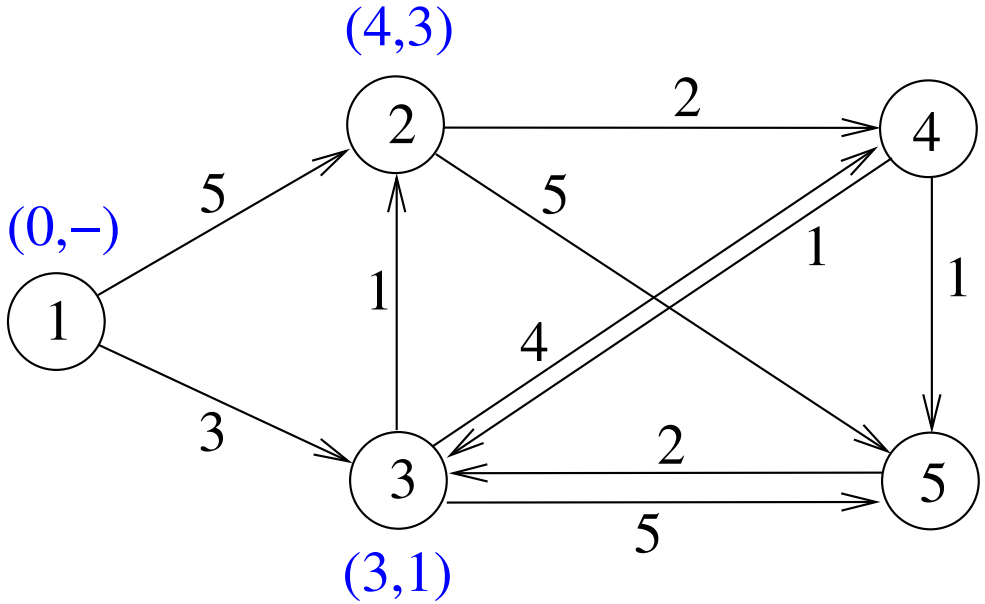
~~3: (3,1)~~

4: (7,3)

5: (8,3)

Nod 2 är bäst av de temporära.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

~~2: (5,1), (4,3)~~

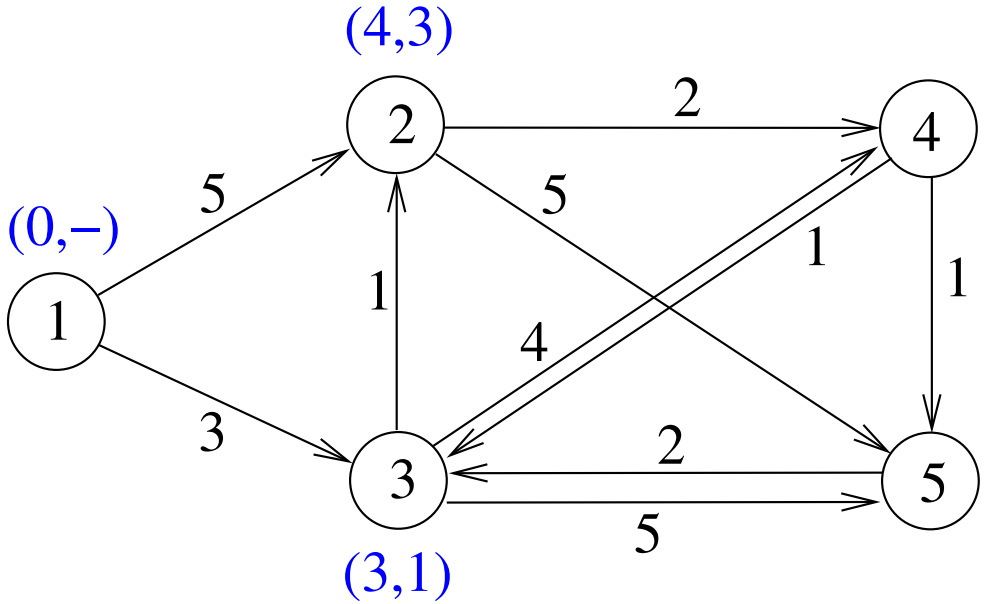
~~3: (3,1)~~

4: ~~(7,3)~~, (6,2)

5: (8,3)

Märk nod 2 permanent. Nod 4 får bättre temp.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

~~2: (5,1), (4,3)~~

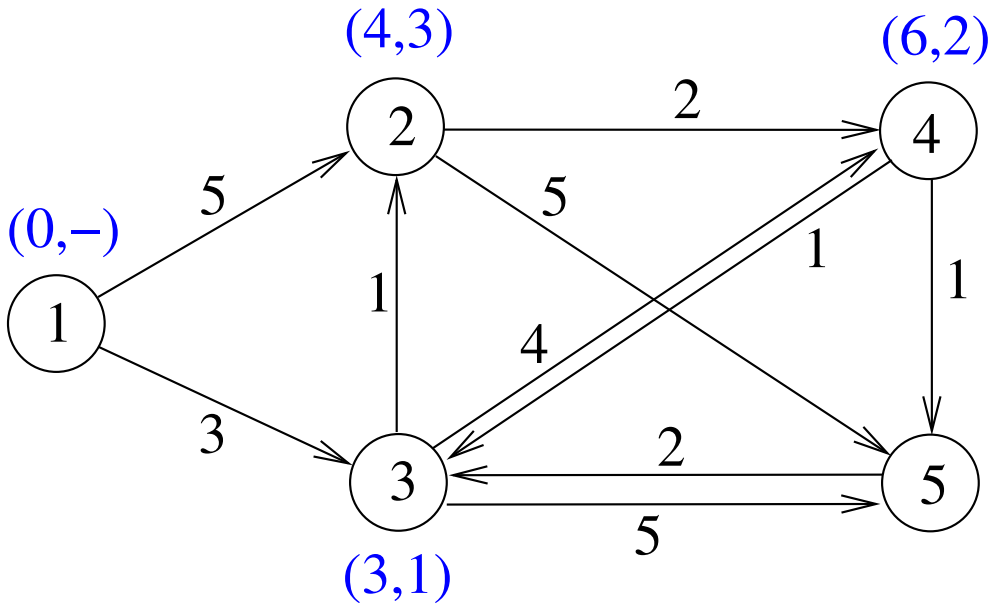
~~3: (3,1)~~

4: ~~(7,3)~~, (6,2)

5: (8,3)

Nod 4 är bäst av de temporära.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

~~2: (5,1), (4,3)~~

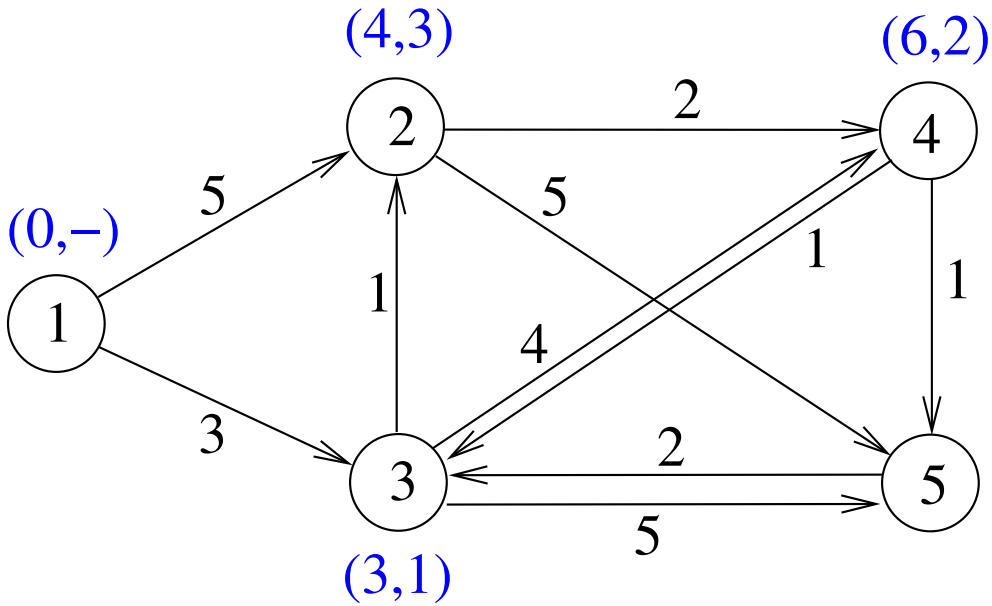
~~3: (3,1)~~

~~4: (7,3), (6,2)~~

5: ~~(8,3)~~, (7,4)

Märk nod 4 permanent. Nod 5 får bättre temp.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

~~2: (5,1), (4,3)~~

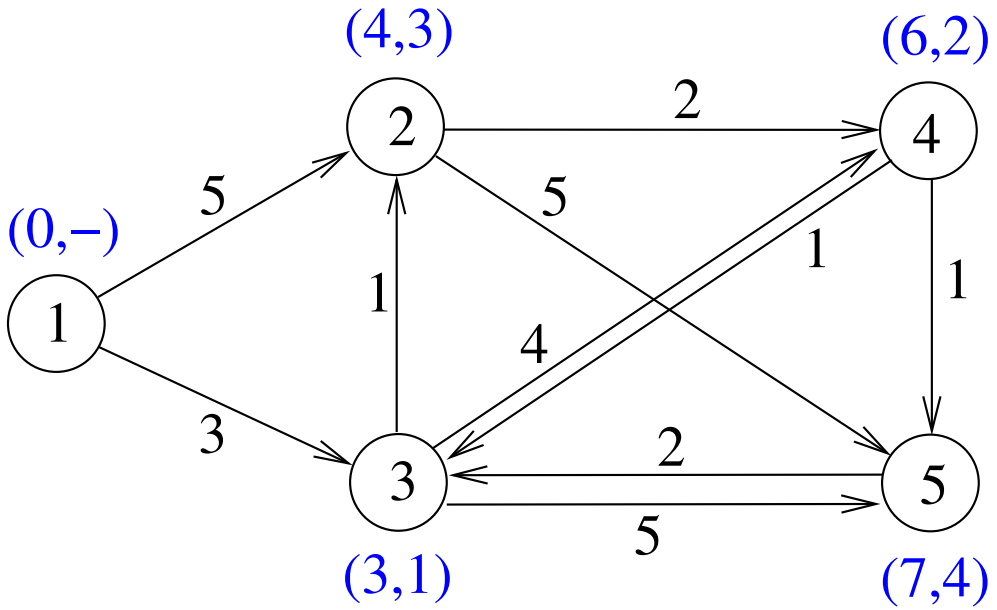
~~3: (3,1)~~

~~4: (7,3), (6,2)~~

5: ~~(8,3)~~, (7,4)

Nod 5 är bäst av de temporära.

Dijkstras metod för billigaste väg: Exempel



temp

~~1: (0,–)~~

~~2: (5,1), (4,3)~~

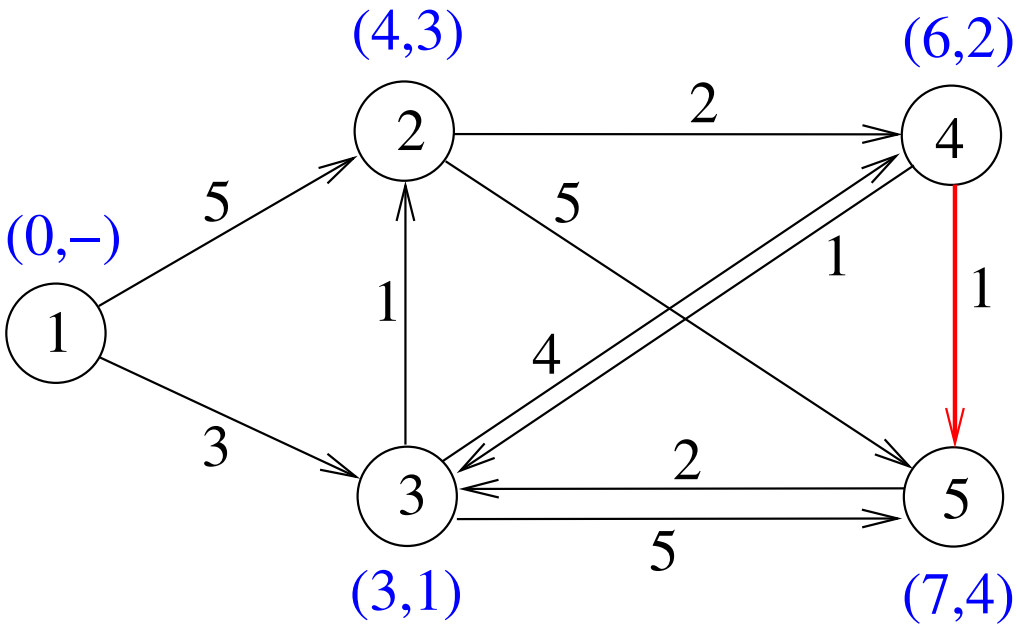
~~3: (3,1)~~

~~4: (7,3), (6,2)~~

~~5: (8,3), (7,4)~~

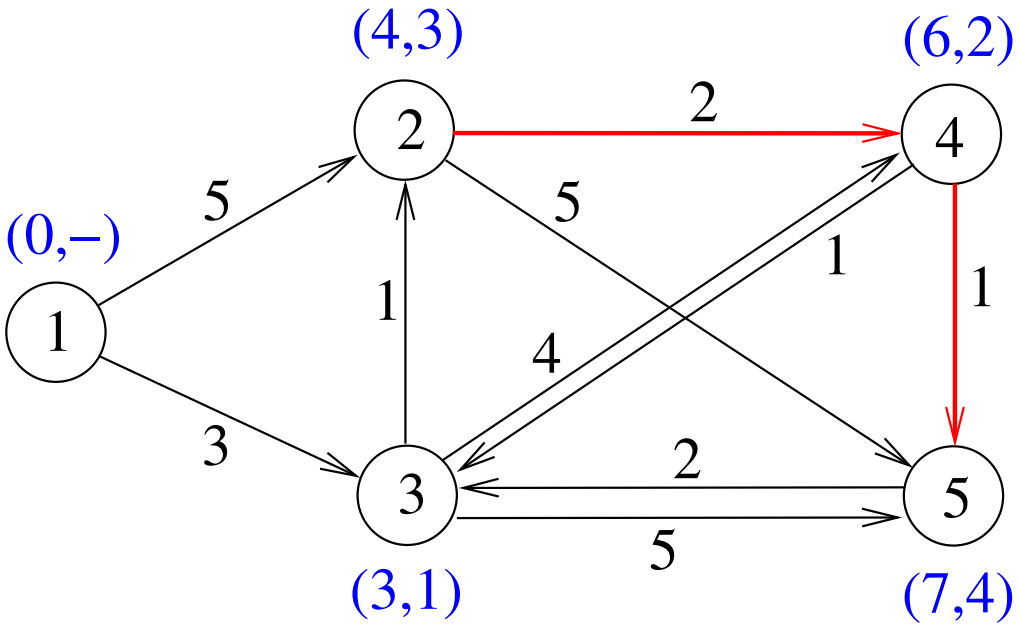
Märk nod 5 permanent. Alla märkta.

Dijkstras metod för billigaste väg: Exempel



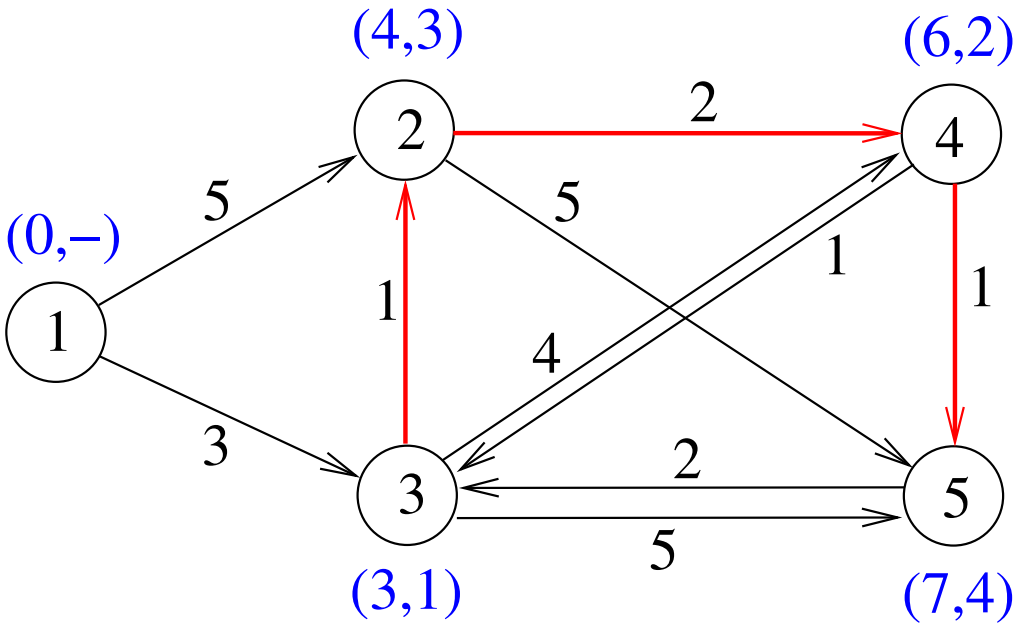
Nysta upp vägen baklänges.

Dijkstras metod för billigaste väg: Exempel



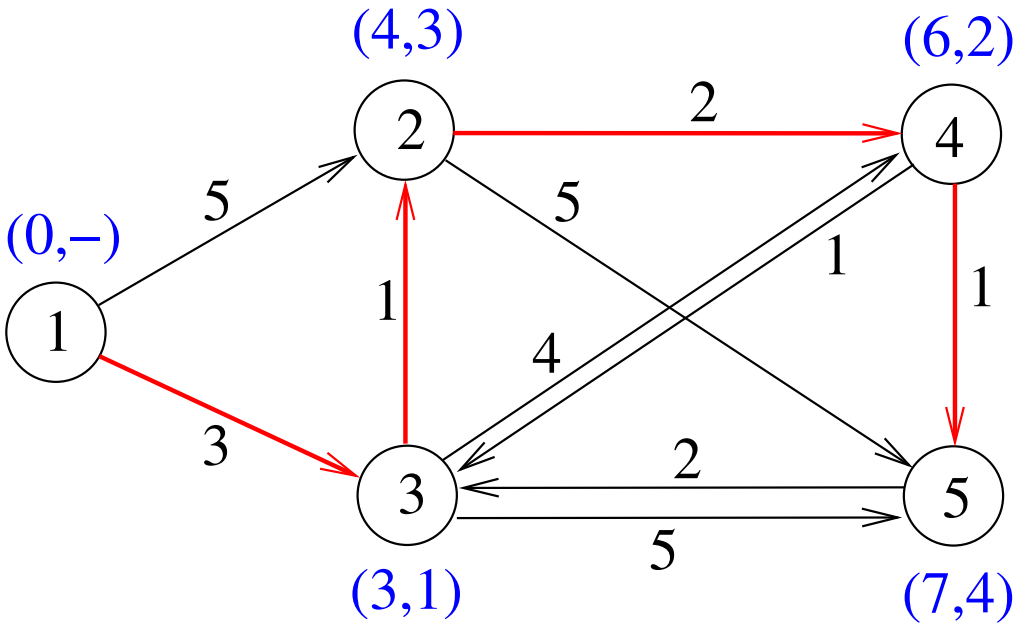
Nysta upp vägen baklänges.

Dijkstras metod för billigaste väg: Exempel



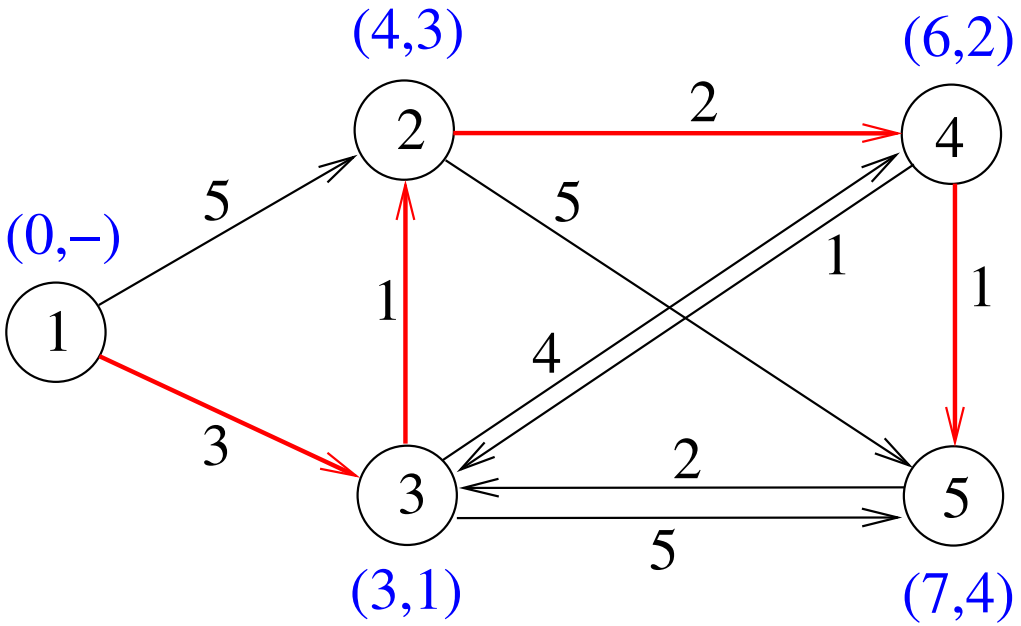
Nysta upp vägen baklänges.

Dijkstras metod för billigaste väg: Exempel



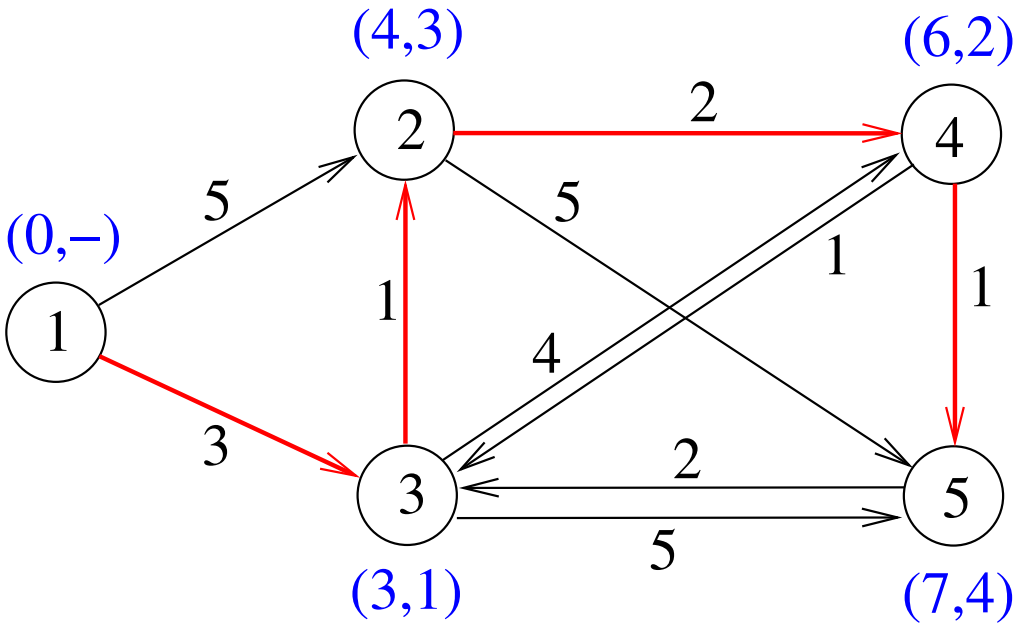
Nysta upp vägen baklänges.

Dijkstras metod för billigaste väg: Exempel



Nysta upp vägen baklänges.
Billigaste väg: 1 - 3 - 2 - 4 - 5. Kostnad: 7.

Dijkstras metod för billigaste väg: Exempel



Nysta upp vägen baklänges.
Billigaste väg: 1 - 3 - 2 - 4 - 5. Kostnad: 7.
Nodpriserna ger optimal duallösning.

Nodmärkningsmetoder, inga cykler

Acyklisk graf:

Nodmärkningsmetoder, inga cykler

Acyklisk graf:

Sortera noderna så att $i < j$ för alla bågar $(i, j) \in B$.

Nodmärkningsmetoder, inga cykler

Acyklisk graf:

Sortera noderna så att $i < j$ för alla bågar $(i, j) \in B$.

Märk noderna i den ordningen: $y_j = \min_{i<j}(c_{ij} + y_i)$ för $j = 1, \ldots, n$.

Nodmärkningsmetoder, inga cykler

Acyklisk graf:

Sortera noderna så att $i < j$ för alla bågar $(i, j) \in B$.

Märk noderna i den ordningen: $y_j = \min_{i<j}(c_{ij} + y_i)$ för $j = 1, \ldots, n$.

(Bellmans ekvationer.

Nodmärkningsmetoder, inga cykler

Acyklisk graf:

Sortera noderna så att $i < j$ för alla bågar $(i, j) \in B$.

Märk noderna i den ordningen: $y_j = \min_{i<j}(c_{ij} + y_i)$ för $j = 1, \ldots, n$.

(Bellmans ekvationer. Används för dynamisk programmering.)

Nodmärkningsmetoder, inga cykler

Acyklisk graf:

Sortera noderna så att $i < j$ för alla bågar $(i, j) \in B$.

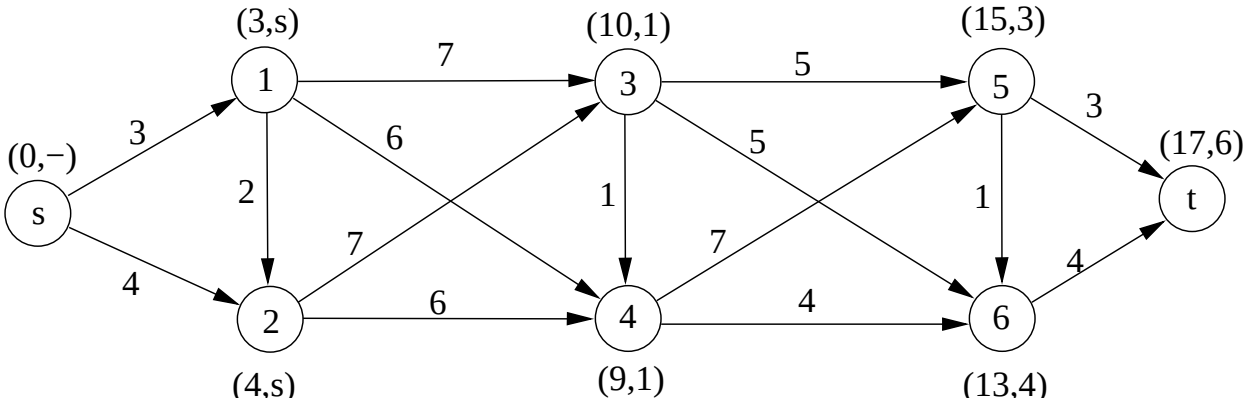
Märk noderna i den ordningen: $y_j = \min_{i<j}(c_{ij} + y_i)$ för $j = 1, \ldots, n$.

(Bellmans ekvationer. Används för dynamisk programmering.)

Man kan aldrig gå baklänges, så inga nodmärkningar behöver göras om.

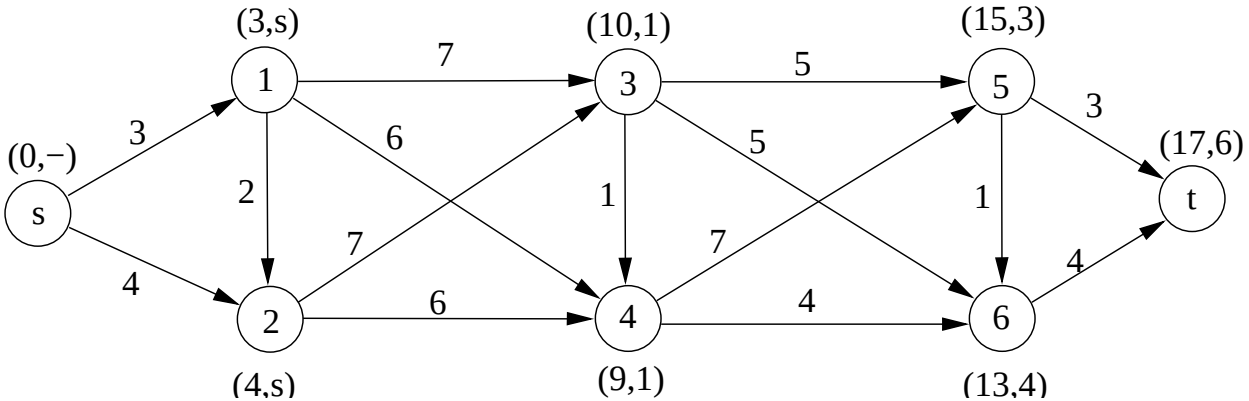
Billigaste väg

Optimala nodmärkningar:



Billigaste väg

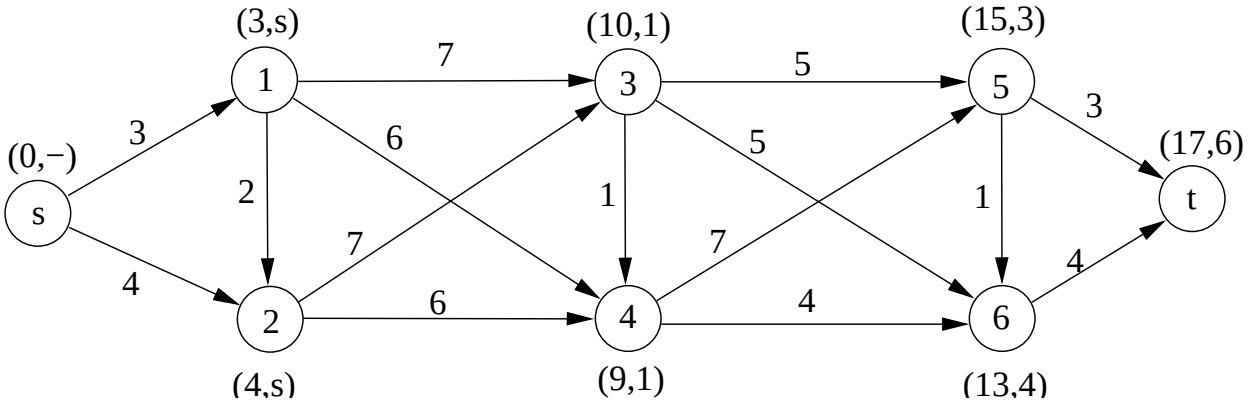
Optimala nodmärkningar:



Alla noder har en märkning.

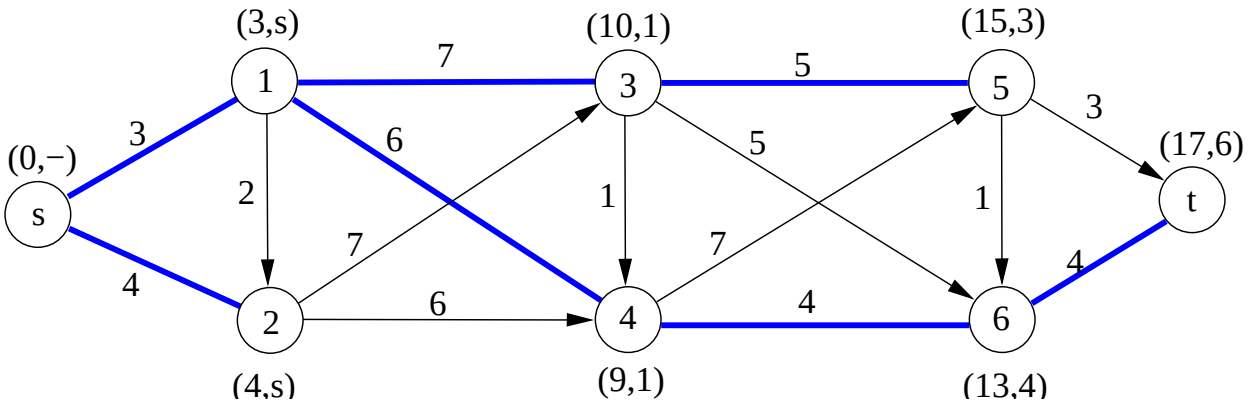
Billigaste väg

Optimala nodmärkningar:



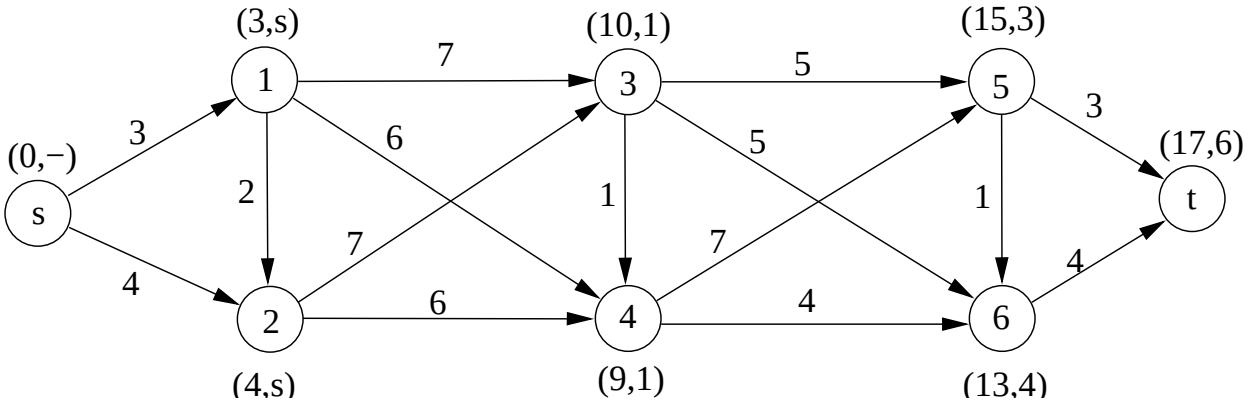
Alla noder har en märkning.

Kan nysta upp från vilken nod som helst.



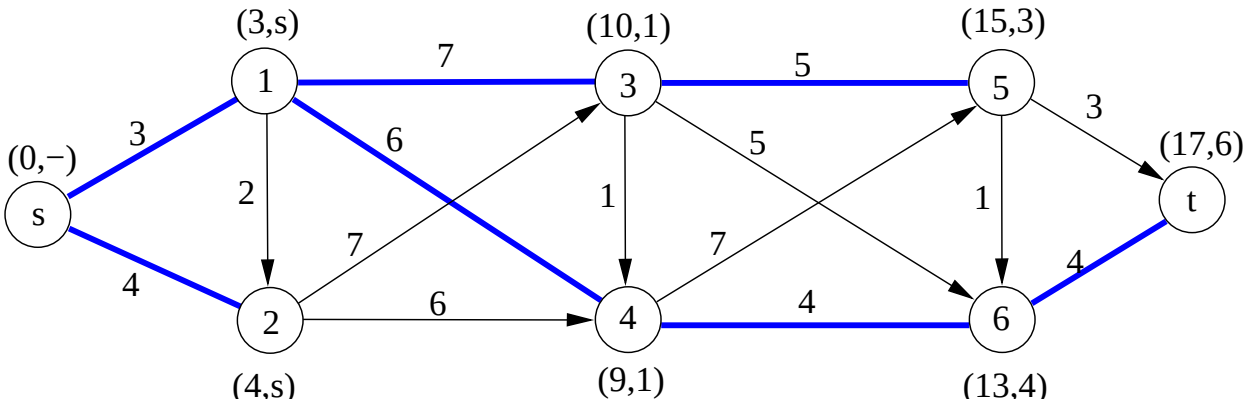
Billigaste väg

Optimala nodmärkningar:



Alla noder har en märkning.

Kan nysta upp från vilken nod som helst.



Vi får billigaste väg från s till alla andra noder.

Billigaste väg: Generaliseringar

Kan lösas med samma metoder.

Billigaste väg	$\min \sum_{ij} c_{ij}$	$y_j = \min_i (c_{ij} + y_i)$	$y_s = 0$

Billigaste väg: Generaliseringar

Kan lösas med samma metoder.

Billigaste väg	$\min \sum_{ij} c_{ij}$	$y_j = \min_i (c_{ij} + y_i)$	$y_s = 0$
Dyraste väg	$\max \sum_{ij} c_{ij}$	$y_j = \max_i (c_{ij} + y_i)$	$y_s = 0$

Billigaste väg: Generaliseringar

Kan lösas med samma metoder.

Billigaste väg	$\min \sum_{ij} c_{ij}$	$y_j = \min_i (c_{ij} + y_i)$	$y_s = 0$
Dyraste väg	$\max \sum_{ij} c_{ij}$	$y_j = \max_i (c_{ij} + y_i)$	$y_s = 0$
Min produkt	$\min \prod_{ij} c_{ij}$	$y_j = \min_i (c_{ij} \cdot y_i)$	$y_s = 1$

Billigaste väg: Generaliseringar

Kan lösas med samma metoder.

Billigaste väg	$\min \sum_{ij} c_{ij}$	$y_j = \min_i (c_{ij} + y_i)$	$y_s = 0$
Dyraste väg	$\max \sum_{ij} c_{ij}$	$y_j = \max_i (c_{ij} + y_i)$	$y_s = 0$
Min produkt	$\min \prod_{ij} c_{ij}$	$y_j = \min_i (c_{ij} \cdot y_i)$	$y_s = 1$
Max produkt	$\max \prod_{ij} c_{ij}$	$y_j = \max_i (c_{ij} \cdot y_i)$	$y_s = 1$

Billigaste väg: Generaliseringar

Kan lösas med samma metoder.

Billigaste väg	$\min \sum_{ij} c_{ij}$	$y_j = \min_i (c_{ij} + y_i)$	$y_s = 0$
Dyraste väg	$\max \sum_{ij} c_{ij}$	$y_j = \max_i (c_{ij} + y_i)$	$y_s = 0$
Min produkt	$\min \prod_{ij} c_{ij}$	$y_j = \min_i (c_{ij} \cdot y_i)$	$y_s = 1$
Max produkt	$\max \prod_{ij} c_{ij}$	$y_j = \max_i (c_{ij} \cdot y_i)$	$y_s = 1$
Min max	$\min \max_{ij} c_{ij}$	$y_j = \min_i (\max(c_{ij}, y_i))$	$y_s = 0$

Billigaste väg: Generaliseringar

Kan lösas med samma metoder.

Billigaste väg	$\min \sum_{ij} c_{ij}$	$y_j = \min_i (c_{ij} + y_i)$	$y_s = 0$
Dyraste väg	$\max \sum_{ij} c_{ij}$	$y_j = \max_i (c_{ij} + y_i)$	$y_s = 0$
Min produkt	$\min \prod_{ij} c_{ij}$	$y_j = \min_i (c_{ij} \cdot y_i)$	$y_s = 1$
Max produkt	$\max \prod_{ij} c_{ij}$	$y_j = \max_i (c_{ij} \cdot y_i)$	$y_s = 1$
Min max	$\min \max_{ij} c_{ij}$	$y_j = \min_i (\max(c_{ij}, y_i))$	$y_s = 0$
Max min	$\max \min_{ij} c_{ij}$	$y_j = \max_i (\min(c_{ij}, y_i))$	$y_s = M$

Billigaste väg: Generaliseringar

Kan lösas med samma metoder.

Billigaste väg	$\min \sum_{ij} c_{ij}$	$y_j = \min_i (c_{ij} + y_i)$	$y_s = 0$
Dyraste väg	$\max \sum_{ij} c_{ij}$	$y_j = \max_i (c_{ij} + y_i)$	$y_s = 0$
Min produkt	$\min \prod_{ij} c_{ij}$	$y_j = \min_i (c_{ij} \cdot y_i)$	$y_s = 1$
Max produkt	$\max \prod_{ij} c_{ij}$	$y_j = \max_i (c_{ij} \cdot y_i)$	$y_s = 1$
Min max	$\min \max_{ij} c_{ij}$	$y_j = \min_i (\max(c_{ij}, y_i))$	$y_s = 0$
Max min	$\max \min_{ij} c_{ij}$	$y_j = \max_i (\min(c_{ij}, y_i))$	$y_s = M$

I de fyra sista fallen förutsätts $c_{ij} \geq 0$ för alla (i, j).

Billigaste väg: Alla till alla

Floyd-Warshalls metod för att finna billigaste väg mellan alla nodpar:

Billigaste väg: Alla till alla

Floyd-Warshalls metod för att finna billigaste väg mellan alla nodpar:

1. Sätt $d_{ij} = c_{ij}$ för alla i, j.
2. För $k = 1, \ldots, |N|$,
   för $i = 1, \ldots, |N|$,
   för $j = 1, \ldots, |N|$
   sätt $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$.

Billigaste väg: Alla till alla

Floyd-Warshalls metod för att finna billigaste väg mellan alla nodpar:

1. Sätt $d_{ij} = c_{ij}$ för alla i, j.
2. För $k = 1, \ldots, |N|$,
   för $i = 1, \ldots, |N|$,
   för $j = 1, \ldots, |N|$
   sätt $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$.

Komplexitet $O(|N|^3)$.

Billigaste väg: Alla till alla

Floyd-Warshalls metod för att finna billigaste väg mellan alla nodpar:

1. Sätt $d_{ij} = c_{ij}$ för alla i, j.
2. För $k = 1, \ldots, |N|$,
   för $i = 1, \ldots, |N|$,
   för $j = 1, \ldots, |N|$
   sätt $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$.

Komplexitet $O(|N|^3)$.

Kan användas för att göra om ett TSPr till ett ΔTSP(r).

Flöde i nätverk

Flöde i nätverk

Variabeldefinition: x_{ij} = flöde i båge (i, j).

Flöde i nätverk

Variabeldefinition: x_{ij} = flöde i båge (i, j).

Bågdata för båge (i, j):

- c_{ij}: flödeskostnad per enhet.
- u_{ij}: övre gräns för flödet.
- l_{ij}: undre gräns för flödet.

Flöde i nätverk

Variabeldefinition: x_{ij} = flöde i båge (i, j).

Bågdata för båge (i, j):

- c_{ij}: flödeskostnad per enhet.
- u_{ij}: övre gräns för flödet.
- l_{ij}: undre gräns för flödet.

Bivillkor: $l_{ij} \leq x_{ij} \leq u_{ij}$

Flöde i nätverk

Variabeldefinition: x_{ij} = flöde i båge (i, j).

Bågdata för båge (i, j):

- c_{ij}: flödeskostnad per enhet.
- u_{ij}: övre gräns för flödet.
- l_{ij}: undre gräns för flödet.

Bivillkor: $l_{ij} \leq x_{ij} \leq u_{ij}$

Noddata för nod i:

- b_i: källstyrka/sänkstyrka.

Flöde i nätverk

Variabeldefinition: x_{ij} = flöde i båge (i, j).

Bågdata för båge (i, j):

- c_{ij}: flödeskostnad per enhet.
- u_{ij}: övre gräns för flödet.
- l_{ij}: undre gräns för flödet.

Bivillkor: $l_{ij} \leq x_{ij} \leq u_{ij}$

Noddata för nod i:

- b_i: källstyrka/sänkstyrka. (måste vara givet)

Flöde i nätverk

Variabeldefinition: x_{ij} = flöde i båge (i, j).

Bågdata för båge (i, j):

- c_{ij}: flödeskostnad per enhet.
- u_{ij}: övre gräns för flödet.
- l_{ij}: undre gräns för flödet.

Bivillkor: $l_{ij} \leq x_{ij} \leq u_{ij}$

Noddata för nod i:

- b_i: källstyrka/sänkstyrka. (måste vara givet)

Nodjämviktsvillkor: $\sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = b_i$ för alla $i \in N$ (in - ut)

Flöde i nätverk

Variabeldefinition: x_{ij} = flöde i båge (i, j).

Bågdata för båge (i, j):

- c_{ij}: flödeskostnad per enhet.
- u_{ij}: övre gräns för flödet.
- l_{ij}: undre gräns för flödet.

Bivillkor: $l_{ij} \leq x_{ij} \leq u_{ij}$

Noddata för nod i:

- b_i: källstyrka/sänkstyrka. (måste vara givet)

Nodjämviktsvillkor: $\sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = b_i$ för alla $i \in N$ (in - ut)

Krav på indata: $\sum_i b_i = 0$.

Flöde i nätverk

Sats

Varje anslutningsmatris är fullständigt unimodulär.

Flöde i nätverk

Sats

Varje anslutningsmatris är fullständigt unimodulär.

Slutsats

Flödesproblem kan betraktas som LP-problem. Flödet blir automatiskt heltal.

Flöde i nätverk

Sats

Varje anslutningsmatris är fullständigt unimodulär.

Slutsats

Flödesproblem kan betraktas som LP-problem. Flödet blir automatiskt heltal.

Obs: Inga andra bivillkor får finnas.

Flöde i nätverk

Sats

Varje anslutningsmatris är fullständigt unimodulär.

Slutsats

Flödesproblem kan betraktas som LP-problem. Flödet blir automatiskt heltal.

Obs: Inga andra bivillkor får finnas.

Minkostnadsflödesproblemet

Flöde i nätverk

Sats

Varje anslutningsmatris är fullständigt unimodulär.

Slutsats

Flödesproblem kan betraktas som LP-problem. Flödet blir automatiskt heltal.

Obs: Inga andra bivillkor får finnas.

Minkostnadsflödesproblemet

Skicka efterfrågade mängder så billigt som möjligt.

Flöde i nätverk

Sats

Varje anslutningsmatris är fullständigt unimodulär.

Slutsats

Flödesproblem kan betraktas som LP-problem. Flödet blir automatiskt heltal.

Obs: Inga andra bivillkor får finnas.

Minkostnadsflödesproblemet

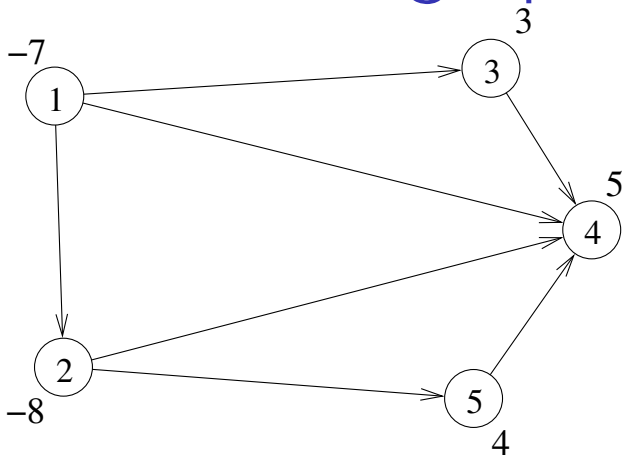
Skicka efterfrågade mängder så billigt som möjligt.

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

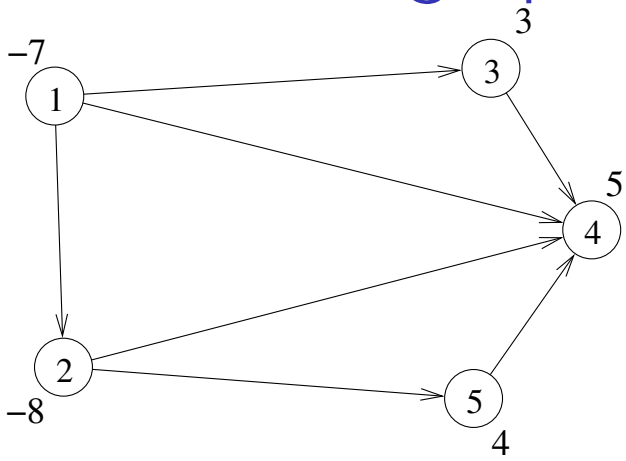
$$\text{då} \quad \sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = b_i \quad \text{för alla } i \in N$$

$$l_{ij} \le x_{ij} \le u_{ij} \quad \text{för alla } (i,j) \in B$$

Nätverksformuleringstips

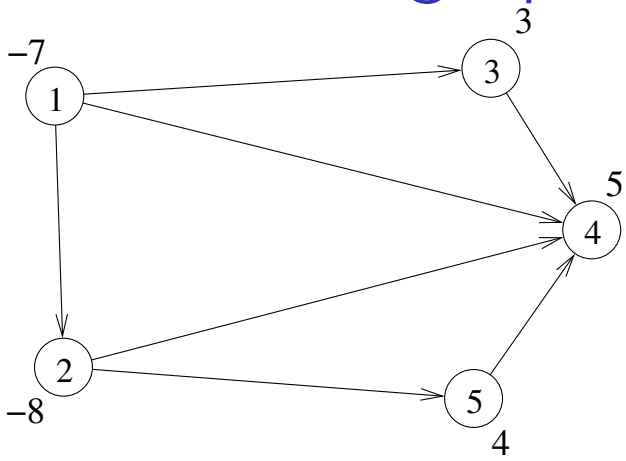


Nätverksformuleringstips



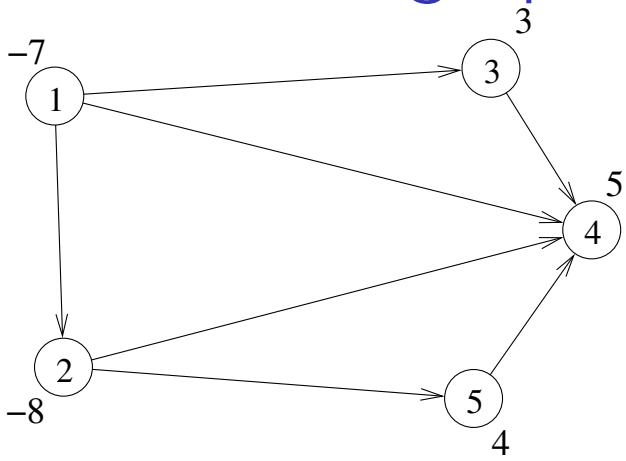
Mer tillgång än efterfrågan:

Nätverksformuleringstips



Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

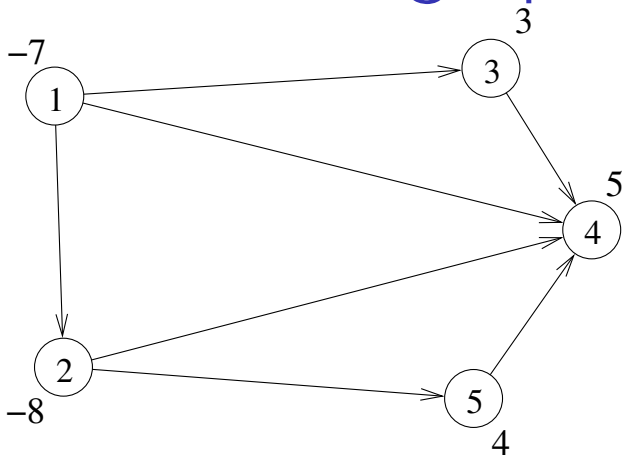
Nätverksformuleringstips



Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

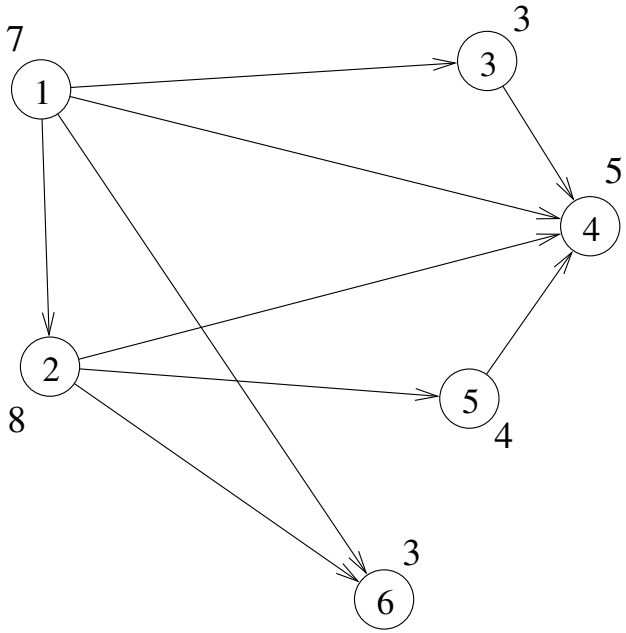
Alternativ 1: Inför dummy-sänka, för varor som egentligen inte skickas.

Nätverksformuleringstips

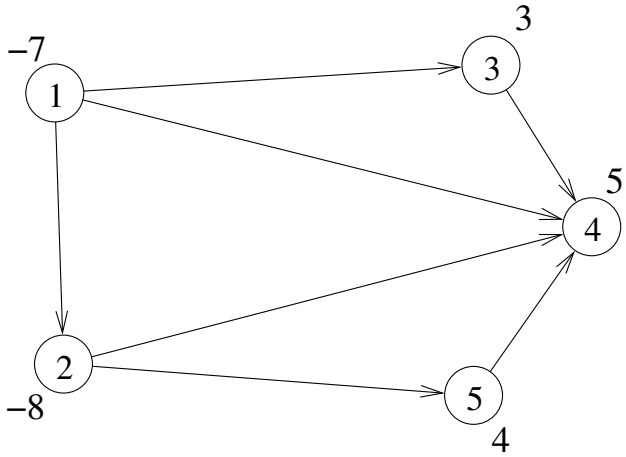


Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

Alternativ 1: Inför dummy-sänka, för varor som egentligen inte skickas.

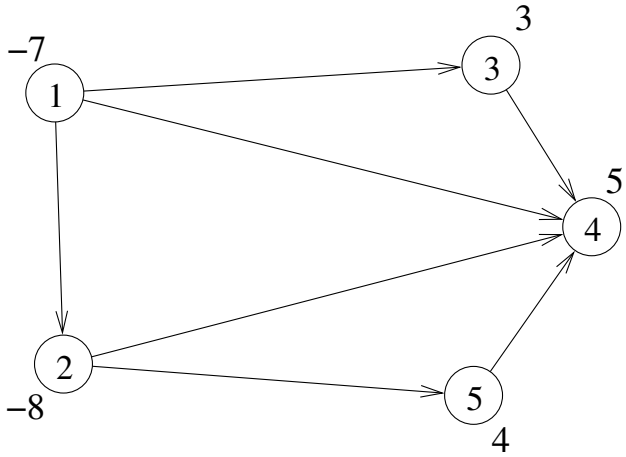


Nätverksformuleringstips



Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

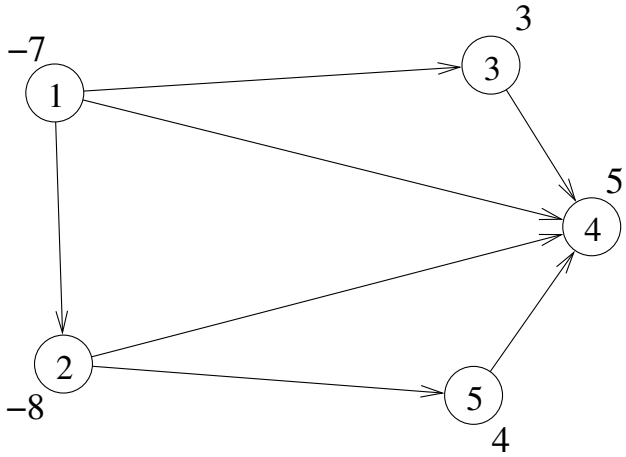
Nätverksformuleringstips



Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

Alternativ 2: Inför superkälla, med tillgång lika med total efterfrågan.

Nätverksformuleringstips

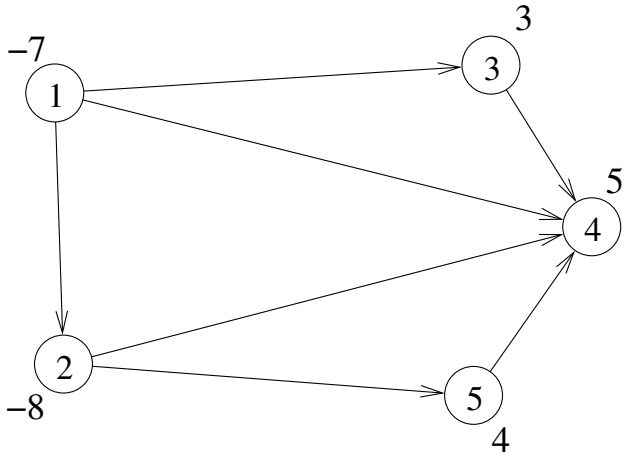


Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

Alternativ 2: Inför superkälla, med tillgång lika med total efterfrågan.

Begränsa flödet i bågar från superkällan till tidigare källor,

Nätverksformuleringstips

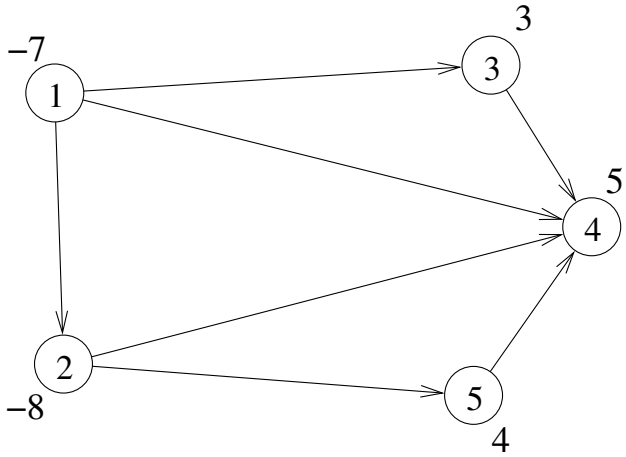


Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

Alternativ 2: Inför superkälla, med tillgång lika med total efterfrågan.

Begränsa flödet i bågar från superkällan till tidigare källor, som nu blir mellannoder.

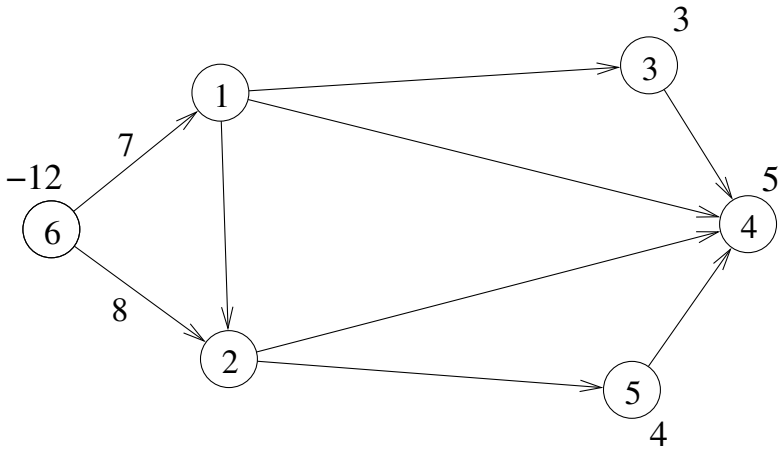
Nätverksformuleringstips



Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

Alternativ 2: Inför superkälla, med tillgång lika med total efterfrågan.

Begränsa flödet i bågar från superkällan till tidigare källor, som nu blir mellannoder.



Nätverksformuleringstips

Köp allt i nod 1 och släng allt i nod 8, men vet inte hur mycket.

Nätverksformuleringstips

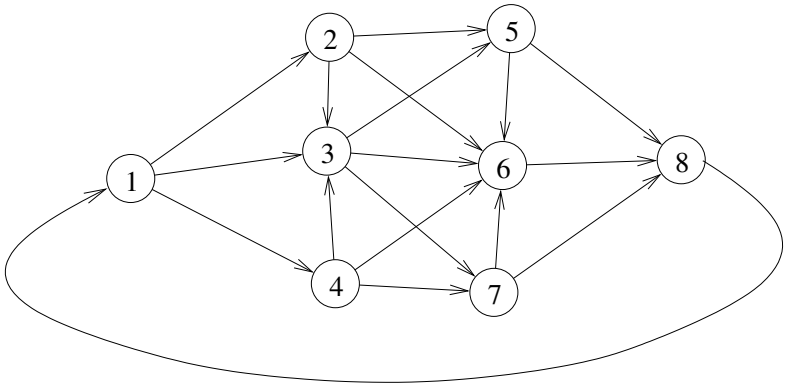
Köp allt i nod 1 och släng allt i nod 8, men vet inte hur mycket.

Alternativ 1: Inför återbåge, nod 1 och 8 mellannoder. Cirkulerande flöde.

Nätverksformuleringstips

Köp allt i nod 1 och släng allt i nod 8, men vet inte hur mycket.

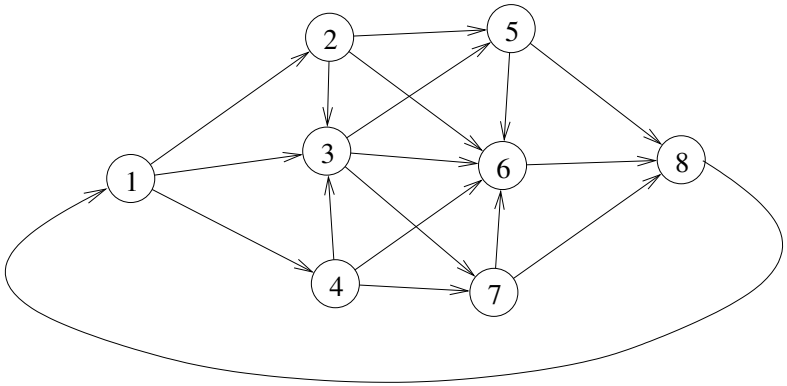
Alternativ 1: Inför återbåge, nod 1 och 8 mellannoder. Cirkulerande flöde.



Nätverksformuleringstips

Köp allt i nod 1 och släng allt i nod 8, men vet inte hur mycket.

Alternativ 1: Inför återbåge, nod 1 och 8 mellannoder. Cirkulerande flöde.

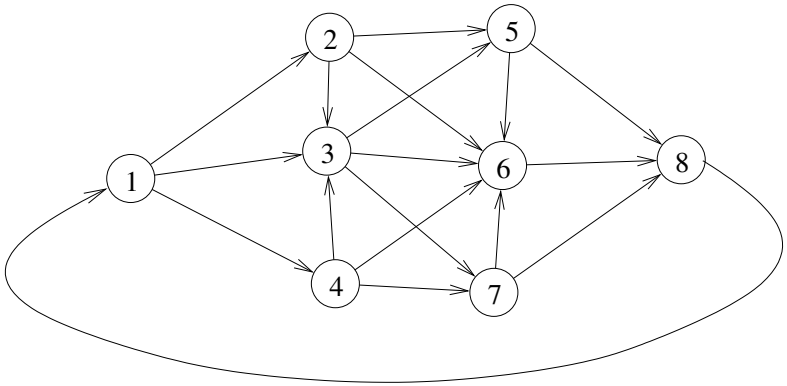


Alternativ 1: Inför slackbåge, nod 1 sänka och nod 8 sänka av styrka 1000.

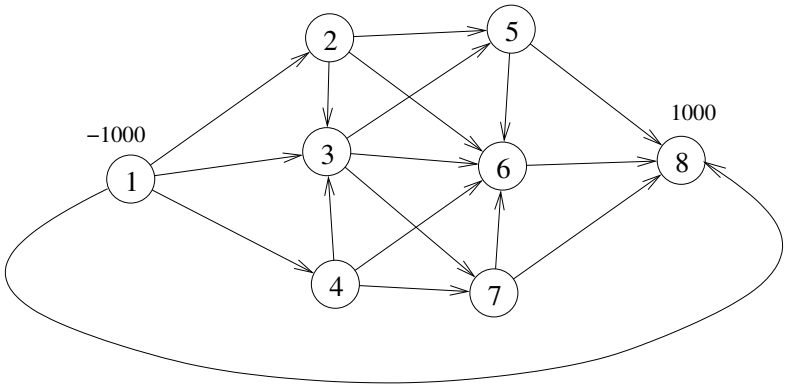
Nätverksformuleringstips

Köp allt i nod 1 och släng allt i nod 8, men vet inte hur mycket.

Alternativ 1: Inför återbåge, nod 1 och 8 mellannoder. Cirkulerande flöde.



Alternativ 1: Inför slackbåge, nod 1 sänka och nod 8 sänka av styrka 1000.



Nätverksformuleringstips

Konvex styckvis linjär kostnad:

Nätverksformuleringstips

Konvex styckvis linjär kostnad:

De först 20 enheterna kostar 10 kr/st, sedan 15 kr/st.

Nätverksformuleringstips

Konvex styckvis linjär kostnad:

De först 20 enheterna kostar 10 kr/st, sedan 15 kr/st.

Inför en parallell båge för det dyrare flödet.

Nätverksformuleringstips

Konvex styckvis linjär kostnad:

De först 20 enheterna kostar 10 kr/st, sedan 15 kr/st.

Inför en parallell båge för det dyrare flödet.

En extranod behövs om parallella bågar inte accepteras.

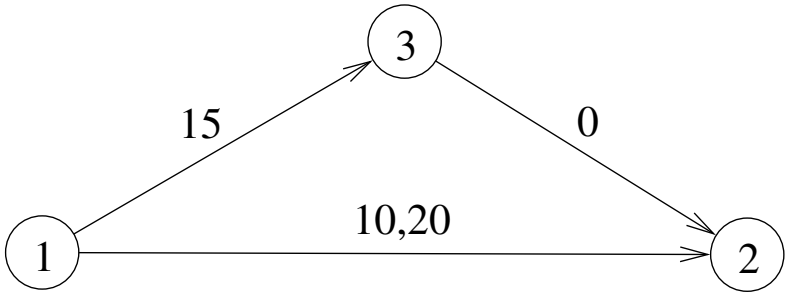
Nätverksformuleringstips

Konvex styckvis linjär kostnad:

De först 20 enheterna kostar 10 kr/st, sedan 15 kr/st.

Inför en parallell båge för det dyrare flödet.

En extranod behövs om parallella bågar inte accepteras.



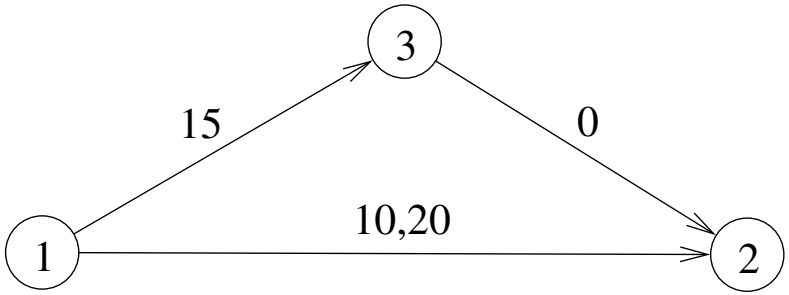
Nätverksformuleringstips

Konvex styckvis linjär kostnad:

De först 20 enheterna kostar 10 kr/st, sedan 15 kr/st.

Inför en parallell båge för det dyrare flödet.

En extranod behövs om parallella bågar inte accepteras.



Observera: Den billigaste bågen kommer automatiskt att användas fullt ut, innan den dyrare börjar användas.

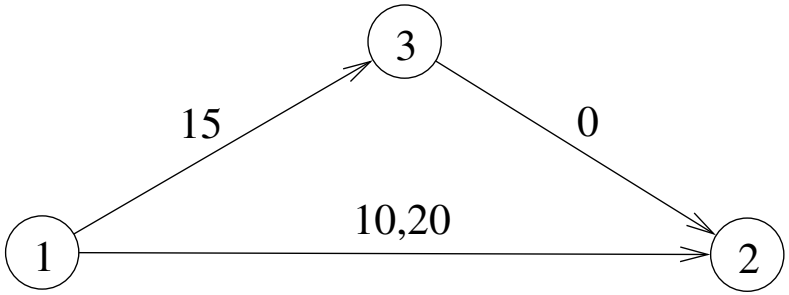
Nätverksformuleringstips

Konvex styckvis linjär kostnad:

De först 20 enheterna kostar 10 kr/st, sedan 15 kr/st.

Inför en parallell båge för det dyrare flödet.

En extranod behövs om parallella bågar inte accepteras.



Observera: Den billigaste bågen kommer automatiskt att användas fullt ut, innan den dyrare börjar användas.

Detta fungerar inte för konkava kostnader.

Vägar: Maxflödesproblemet

Hur mycket kan man maximalt få igenom en fabrik med många stationer,

Vägar: Maxflödesproblemet

Hur mycket kan man maximalt få igenom en fabrik med många stationer, och vilka stationer är det som är begränsande?

Vägar: Maxflödesproblemet

Hur mycket kan man maximalt få igenom en fabrik med många stationer, och vilka stationer är det som är begränsande?

Indata: Riktad graf $G = (N, B)$ med bågkapaciteter u, start/slutnod s, t.

Vägar: Maxflödesproblemet

Hur mycket kan man maximalt få igenom en fabrik med många stationer, och vilka stationer är det som är begränsande?

Indata: Riktad graf $G = (N, B)$ med bågkapaciteter u, start/slutnod s, t.

Skicka så mycket som möjligt från s till t.

Vägar: Maxflödesproblemet

Hur mycket kan man maximalt få igenom en fabrik med många stationer, och vilka stationer är det som är begränsande?
Indata: Riktad graf $G = (N, B)$ med bågkapaciteter u, start/slutnod s, t.

Skicka så mycket som möjligt från s till t.

Variabeldefinition: x_{ij} = flöde i båge (i, j).

Vägar: Maxflödesproblemet

Hur mycket kan man maximalt få igenom en fabrik med många stationer, och vilka stationer är det som är begränsande?

Indata: Riktad graf $G = (N, B)$ med bågkapaciteter u, start/slutnod s, t.

Skicka så mycket som möjligt från s till t.

Variabeldefinition: x_{ij} = flöde i båge (i, j).

$$\max \quad f$$

$$\text{då} \quad \sum_{j:(j,i)\in B} x_{ji} - \sum_{j:(i,j)\in B} x_{ij} = \begin{cases} -f & \text{då } i = s \\ f & \text{då } i = t \\ 0 & \text{f ö} \end{cases} \quad \text{för alla } i \in N$$

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

$$f \text{ fri}$$

Lösningsmetod för maxflödesproblemet

Påfyllnadsmetoden (Edmonds-Karp)

Lösningsmetod för maxflödesproblemet

Påfyllnadsmetoden (Edmonds-Karp)

0. Börja från noll.

Lösningsmetod för maxflödesproblemet

Påfyllnadsmetoden (Edmonds-Karp)

0. Börja från noll.
1. Finn maximal **flödesökande väg** från s till t. Avbryt om ingen väg finns.

Lösningsmetod för maxflödesproblemet

Påfyllnadsmetoden (Edmonds-Karp)

0. Börja från noll.
1. Finn maximal **flödesökande väg** från s till t. Avbryt om ingen väg finns.
2. Skicka så mycket som möjligt den vägen.

Lösningsmetod för maxflödesproblemet

Påfyllnadsmetoden (Edmonds-Karp)

0. Börja från noll.
1. Finn maximal **flödesökande väg** från s till t. Avbryt om ingen väg finns.
2. Skicka så mycket som möjligt den vägen.
3. Ändra tillåtna riktningar.

Lösningsmetod för maxflödesproblemet

Påfyllnadsmetoden (Edmonds-Karp)

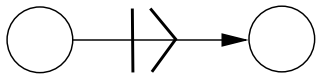
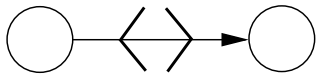
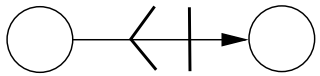
0. Börja från noll.
1. Finn maximal **flödesökande väg** från s till t. Avbryt om ingen väg finns.
2. Skicka så mycket som möjligt den vägen.
3. Ändra tillåtna riktningar.
4. Gå till 1.

Lösningsmetod för maxflödesproblemet

Påfyllnadsmetoden (Edmonds-Karp)

0. Börja från noll.
1. Finn maximal **flödesökande väg** från s till t. Avbryt om ingen väg finns.
2. Skicka så mycket som möjligt den vägen.
3. Ändra tillåtna riktningar.
4. Gå till 1.

Tillåtna riktningar:

$x_{ij} = 0$:	Framåt (öka).	
$0 < x_{ij} < u_{ij}$:	Framåt och bakåt.	
$x_{ij} = u_{ij}$:	Bakåt (minska).	

Maxflödesproblemet

Sats

Varje tillåtet flöde ger en *undre* gräns på f^*.
Kapaciteten hos varje (s, t)-snitt ger en *övre* gräns på f^*.

Maxflödesproblemet

Sats

Varje tillåtet flöde ger en *undre* gräns på f^*.
Kapaciteten hos varje (s, t)-snitt ger en *övre* gräns på f^*.

Sats

Kapaciteten hos ett *minsnitt* är lika med den maximala flödesstyrkan, f^*.

Maxflödesproblemet

Sats

Varje tillåtet flöde ger en *undre* gräns på f^*.
Kapaciteten hos varje (s, t)-snitt ger en *övre* gräns på f^*.

Sats

Kapaciteten hos ett *minsnitt* är lika med den maximala flödesstyrkan, f^*.

Sats

Flödesstyrkan är maximal (och lika med kapaciteten hos ett minsnitt) om och endast om en flödesökande väg saknas.

Maxflödesproblemet

Sats

Varje tillåtet flöde ger en *undre* gräns på f^*.
Kapaciteten hos varje (s, t)-snitt ger en *övre* gräns på f^*.

Sats

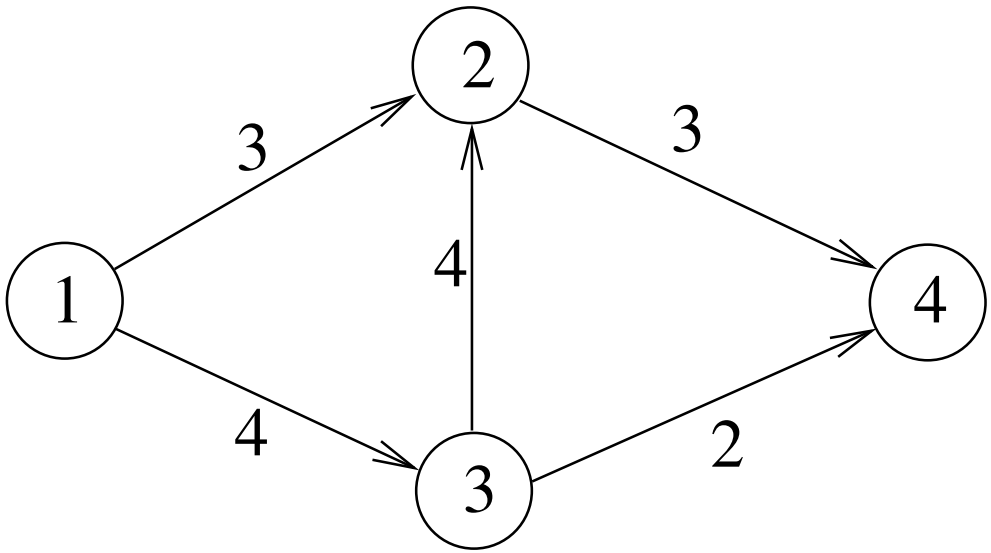
Kapaciteten hos ett *minsnitt* är lika med den maximala flödesstyrkan, f^*.

Sats

Flödesstyrkan är maximal (och lika med kapaciteten hos ett minsnitt) om och endast om en flödesökande väg saknas.

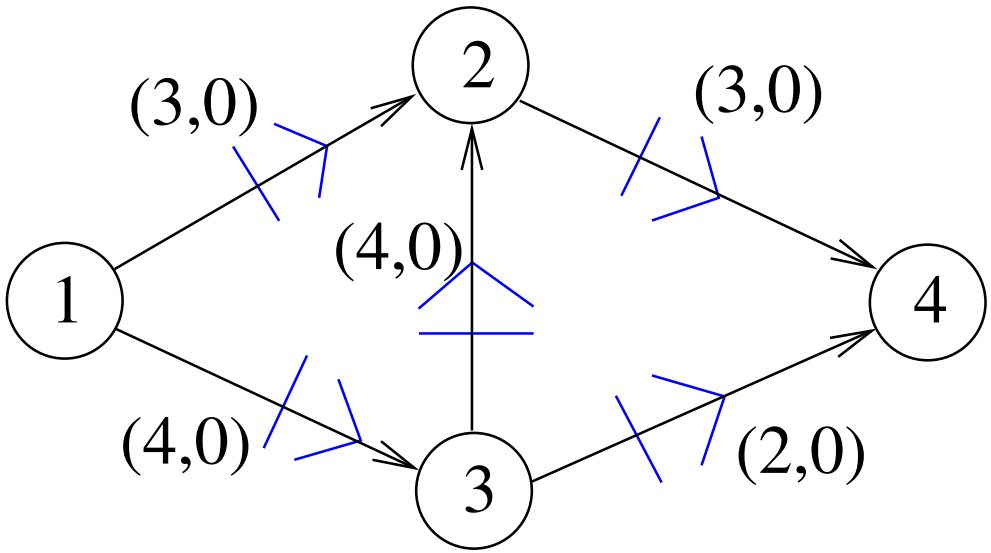
En flödesökande väg finnes metodiskt med Dijkstras metod (max av min).

Maxflöde: Exempel



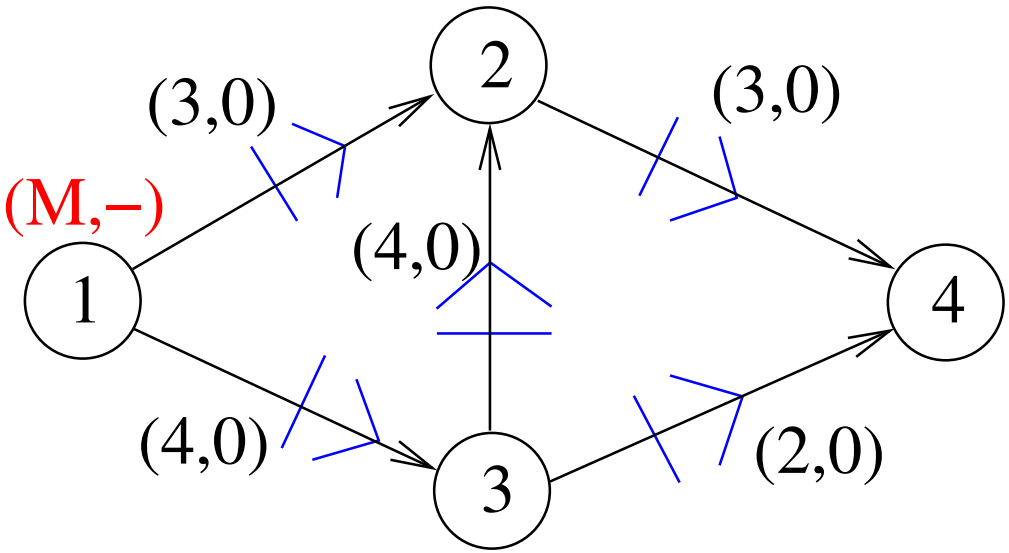
Graf med kapaciteter.

Maxflöde: Exempel



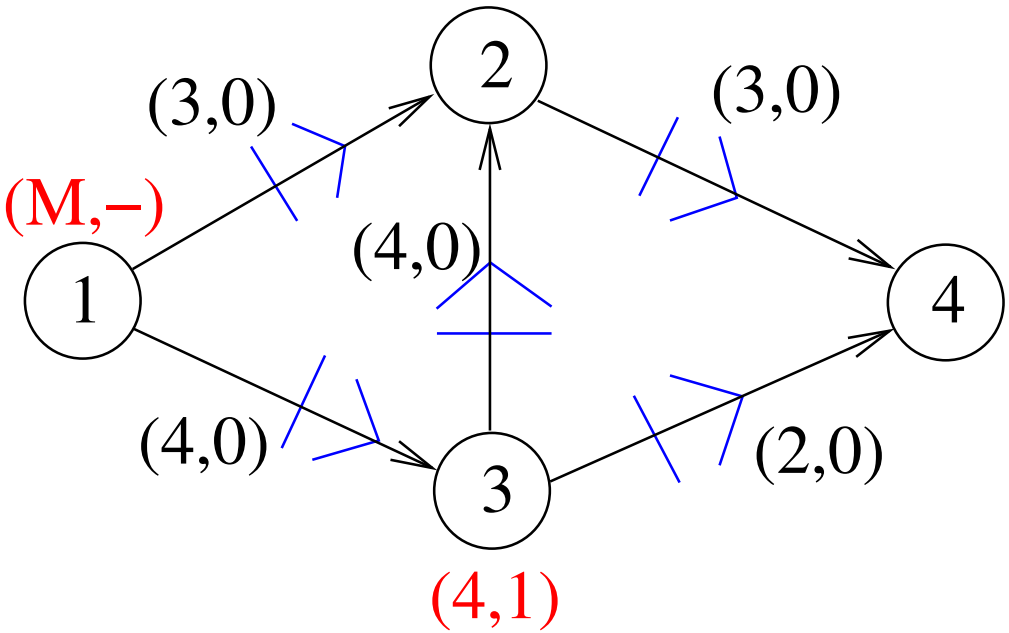
Markera tillåtna riktningar.

Maxflöde: Exempel



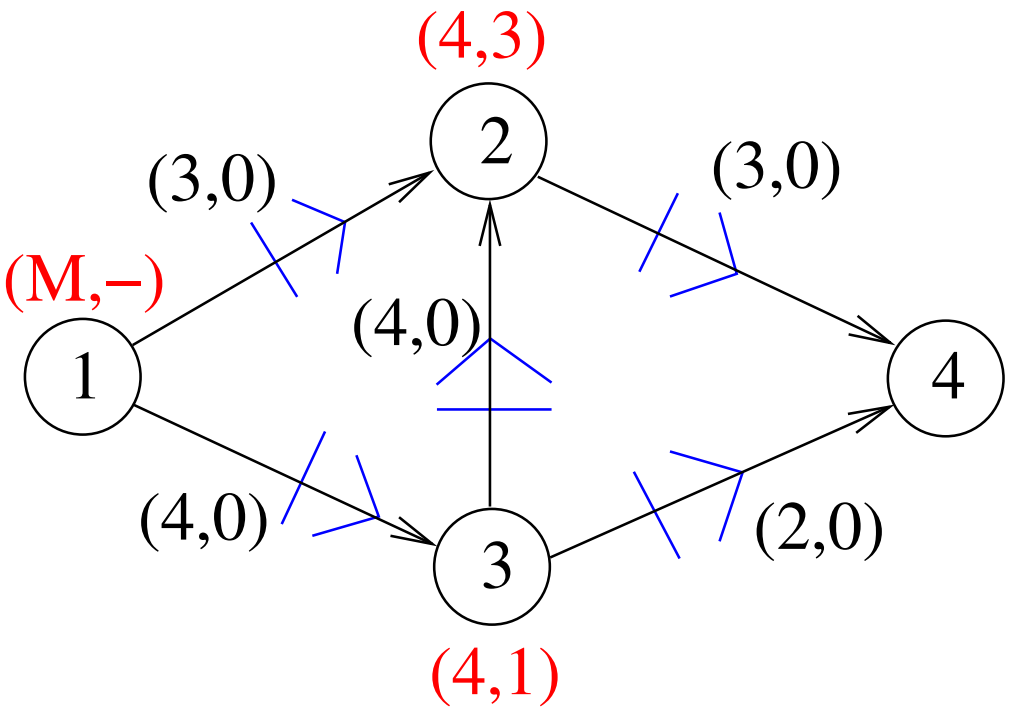
Sök väg med Dijkstras metod.

Maxflöde: Exempel



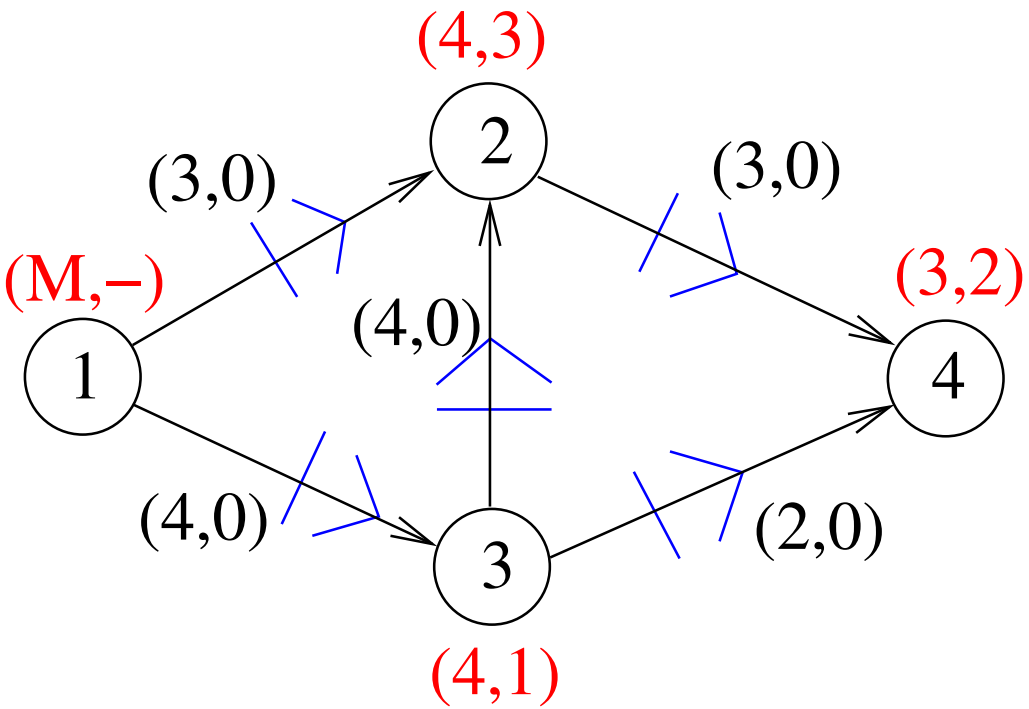
Sök väg med Dijkstras metod.

Maxflöde: Exempel



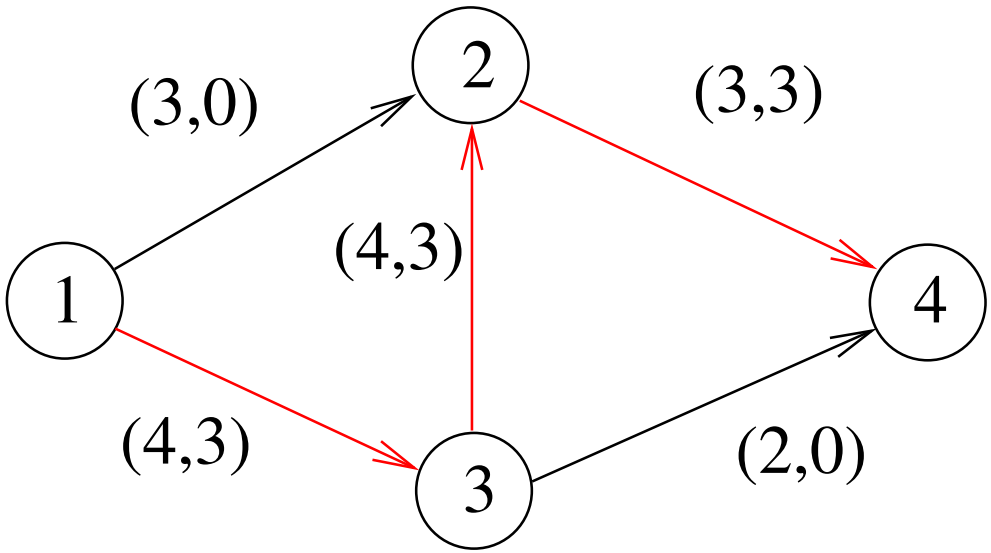
Sök väg med Dijkstras metod.

Maxflöde: Exempel



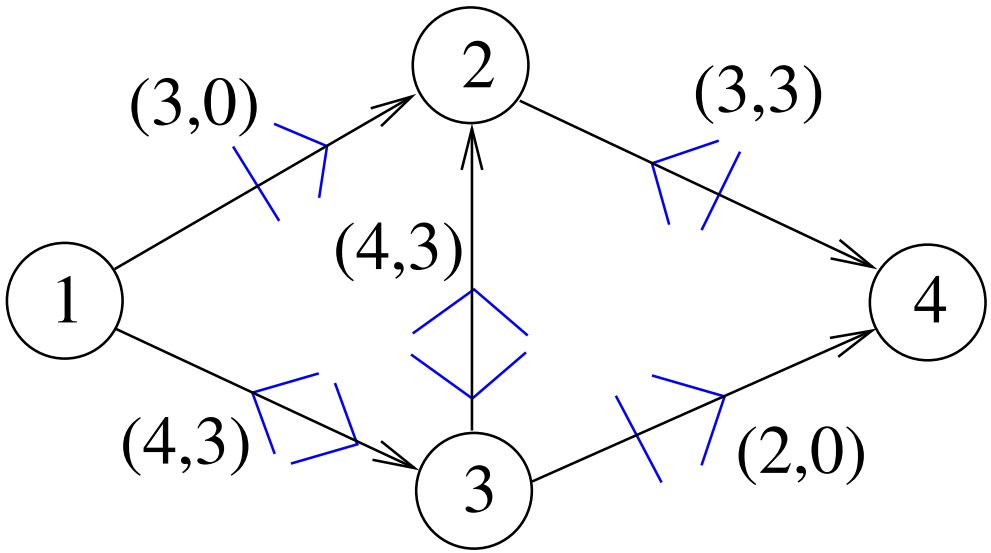
Sök väg med Dijkstras metod.

Maxflöde: Exempel



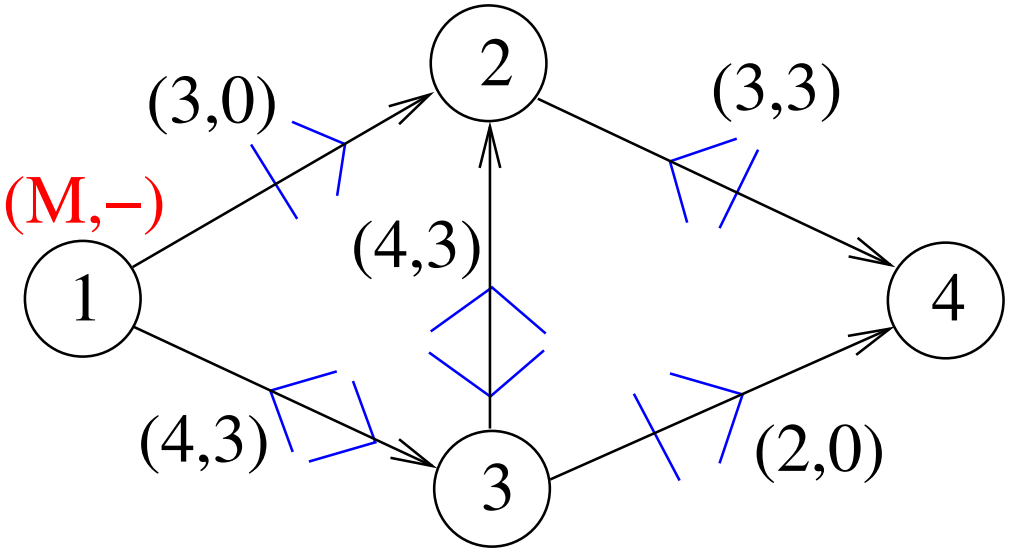
Väg med kapacitet 3 funnen. Skicka 3 enheter.

Maxflöde: Exempel



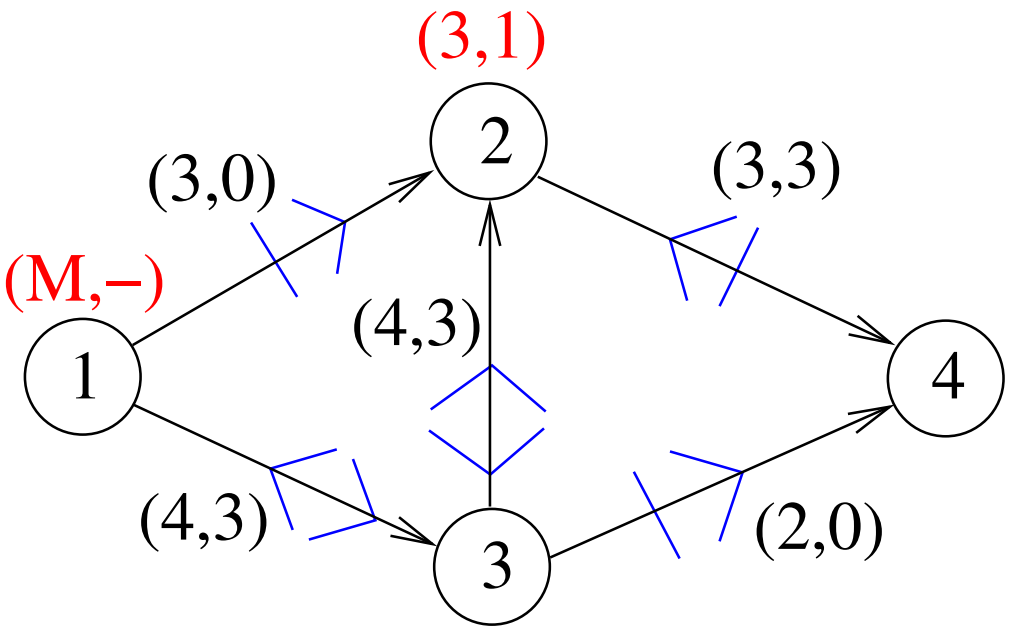
Markera tillåtna riktningar.

Maxflöde: Exempel



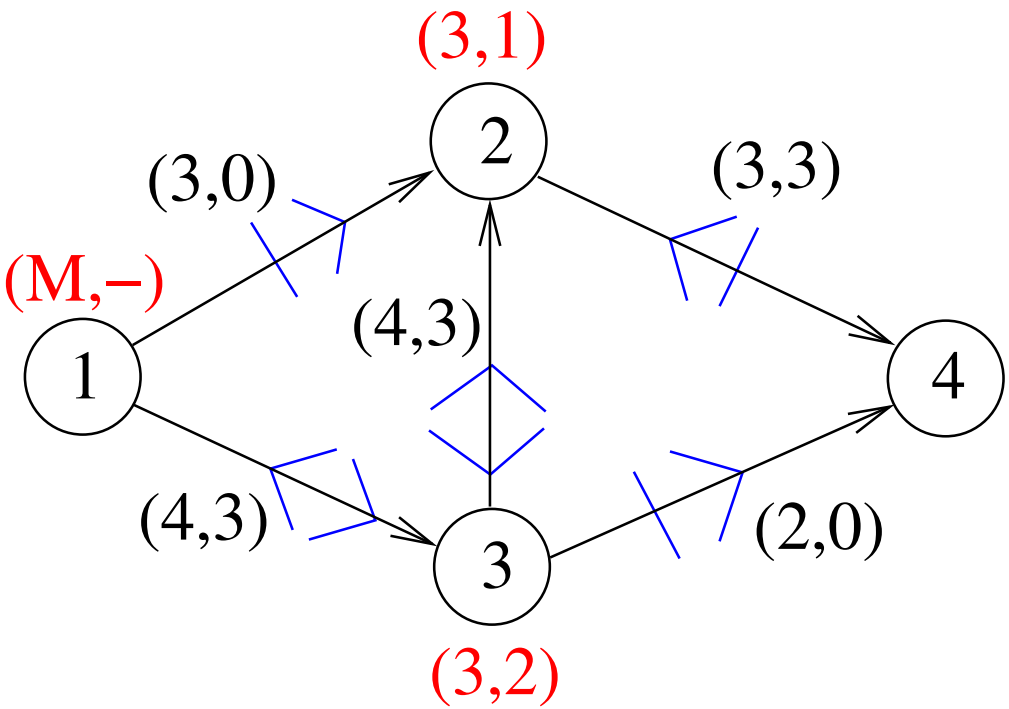
Sök väg med Dijkstras metod.

Maxflöde: Exempel



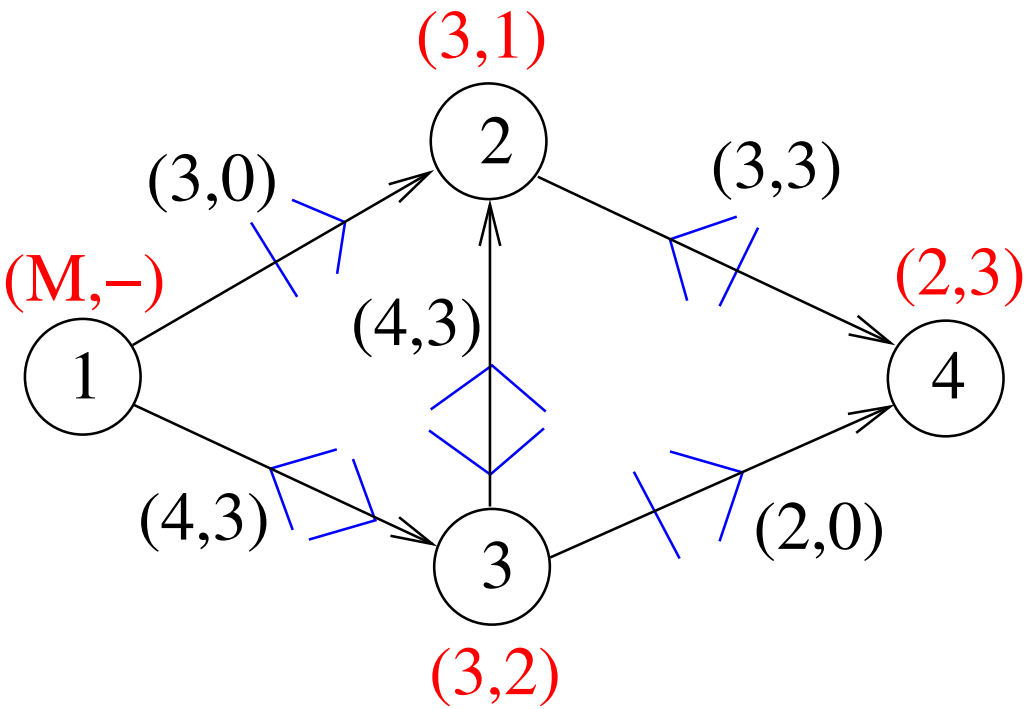
Sök väg med Dijkstras metod. (Får minska flöde.)

Maxflöde: Exempel



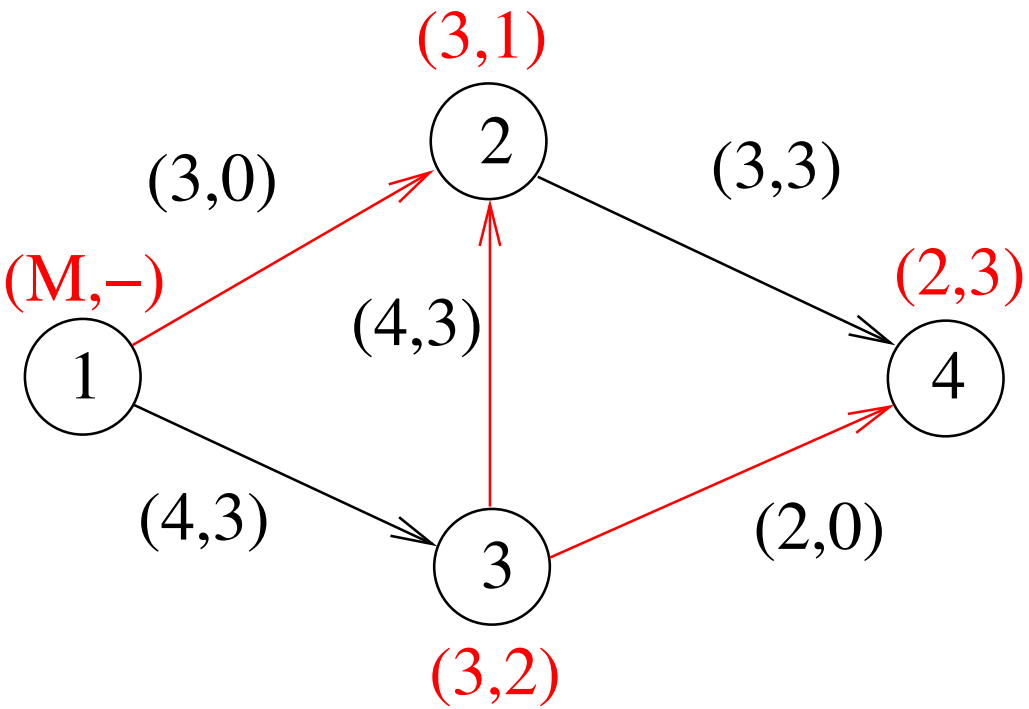
Sök väg med Dijkstras metod.

Maxflöde: Exempel



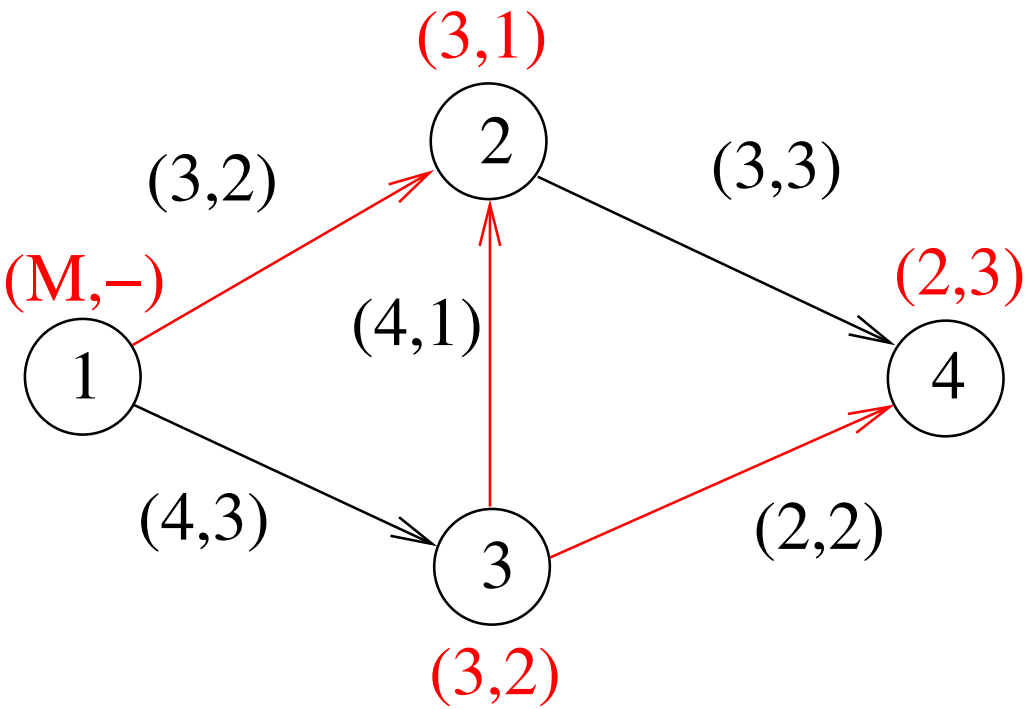
Sök väg med Dijkstras metod.

Maxflöde: Exempel



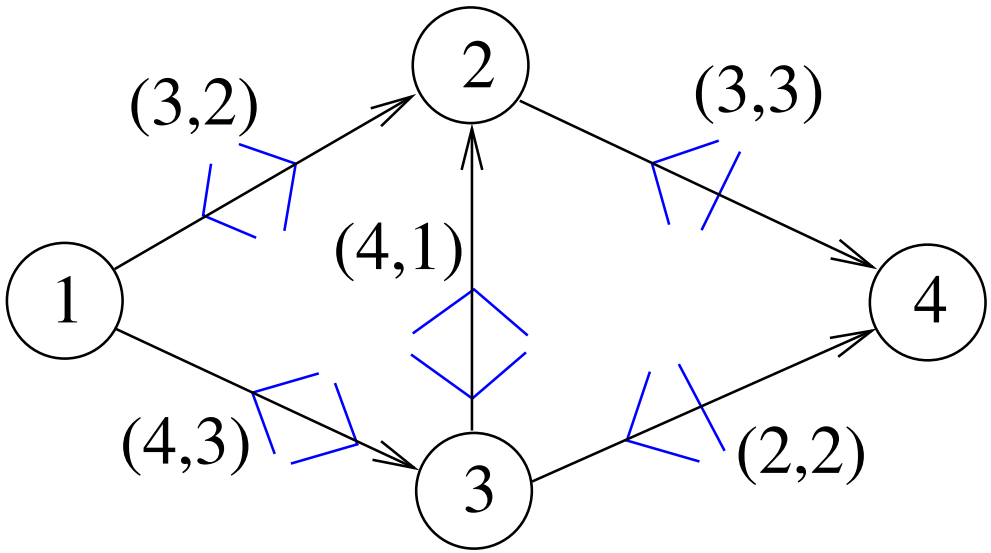
Väg med kapacitet 2 funnen.

Maxflöde: Exempel



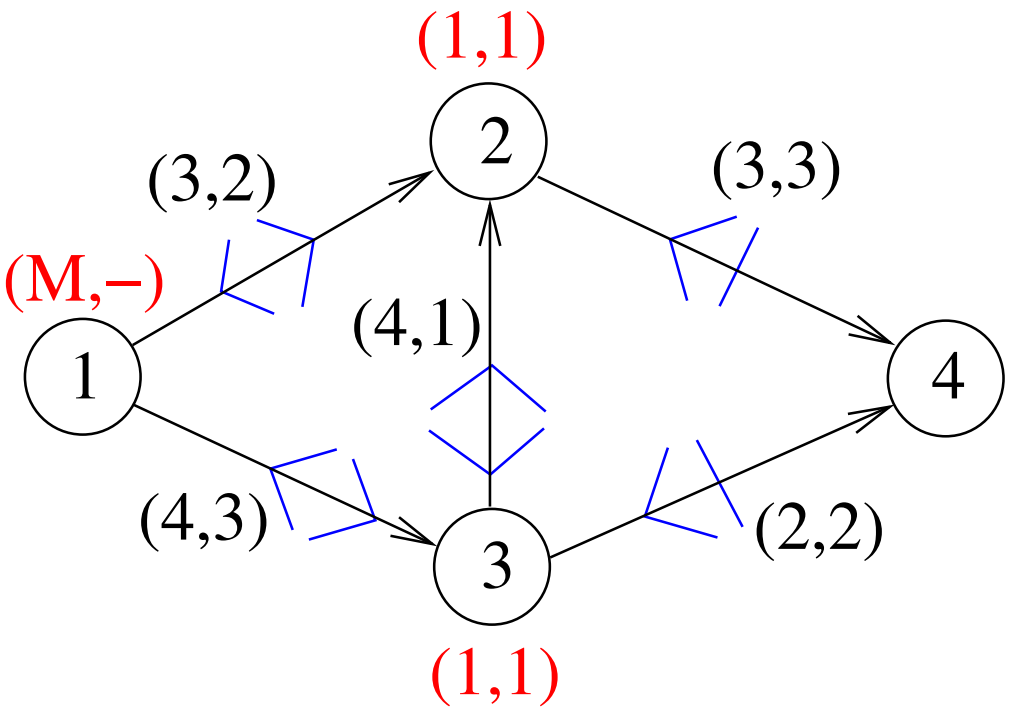
Skicka 2 enheter.

Maxflöde: Exempel



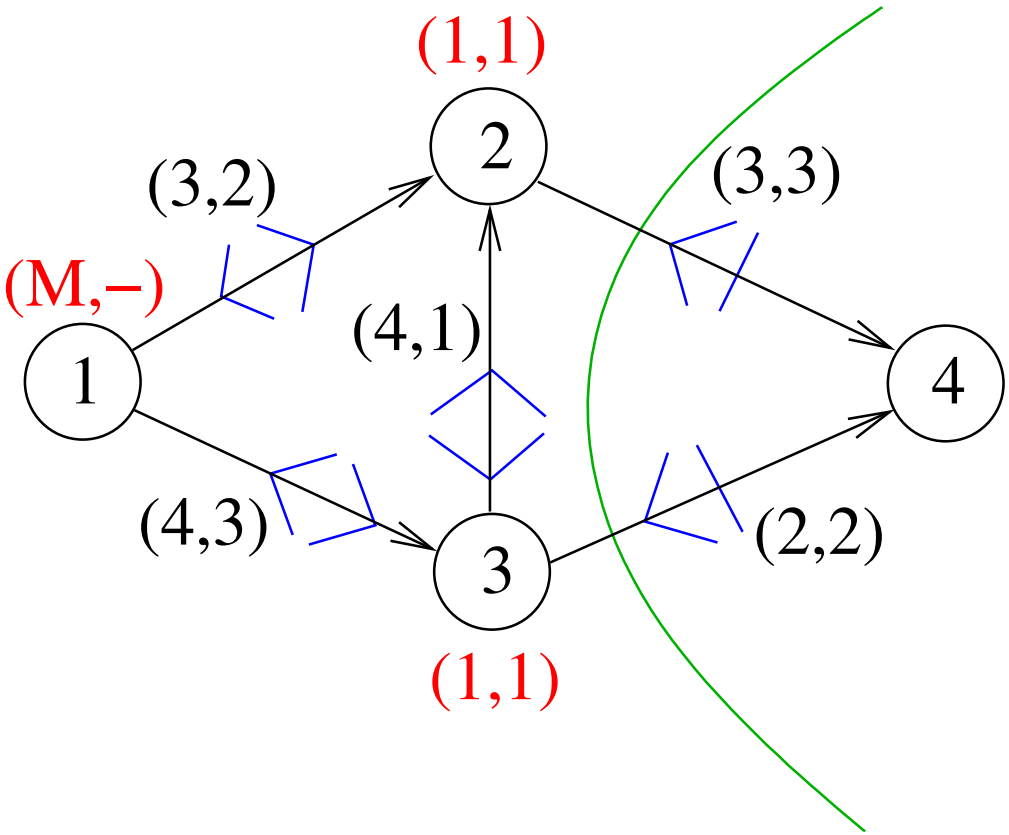
Markera tillåtna riktningar.

Maxflöde: Exempel



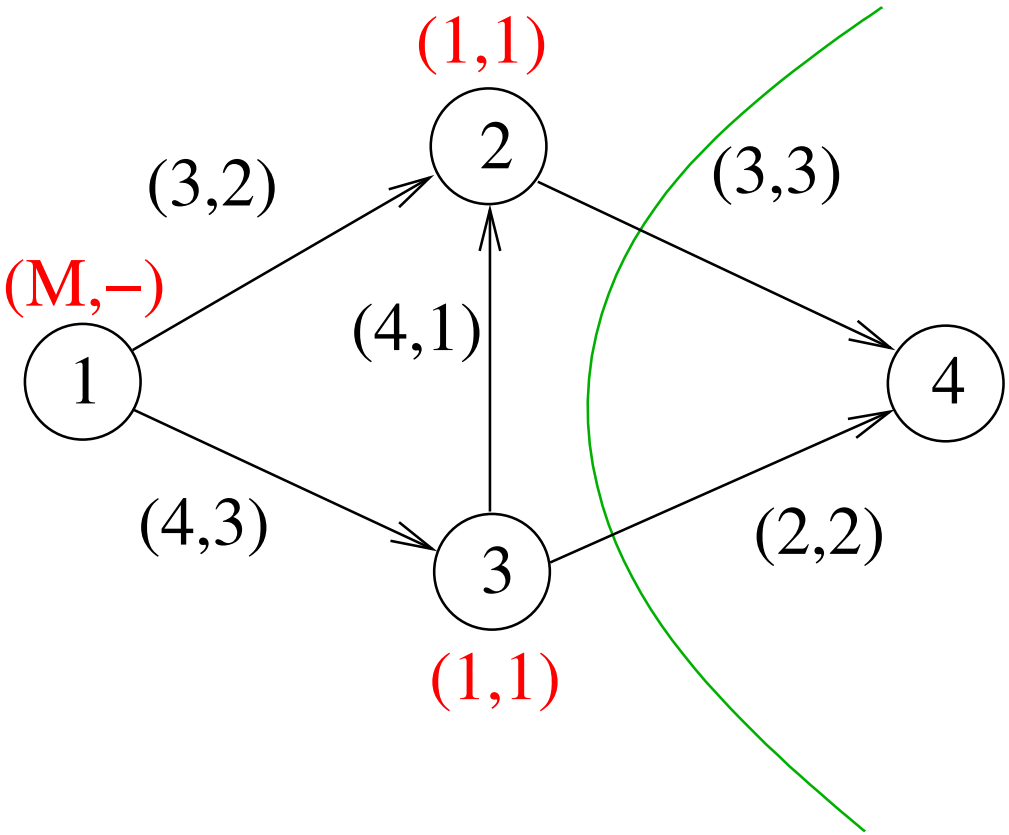
Sök väg med Dijkstras metod.

Maxflöde: Exempel



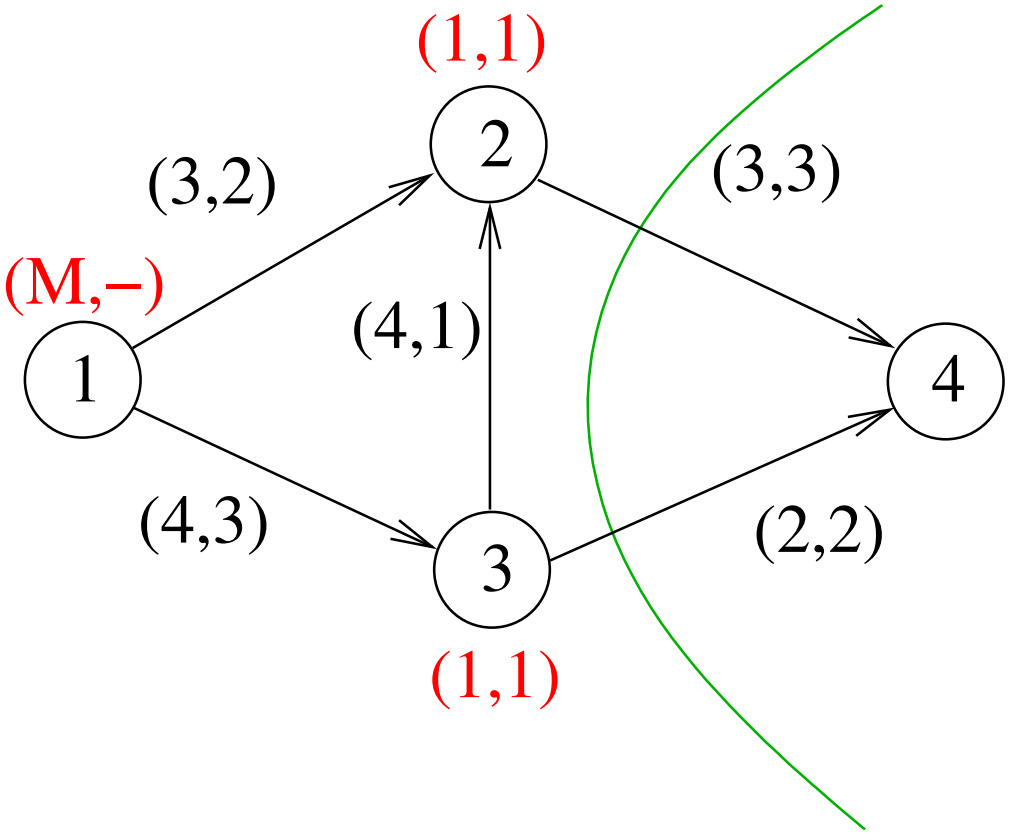
Sök väg med Dijkstras metod. Stopp.

Maxflöde: Exempel



Maxflöde uppnått. Minsnittet går mellan märkta och omärkta noder.

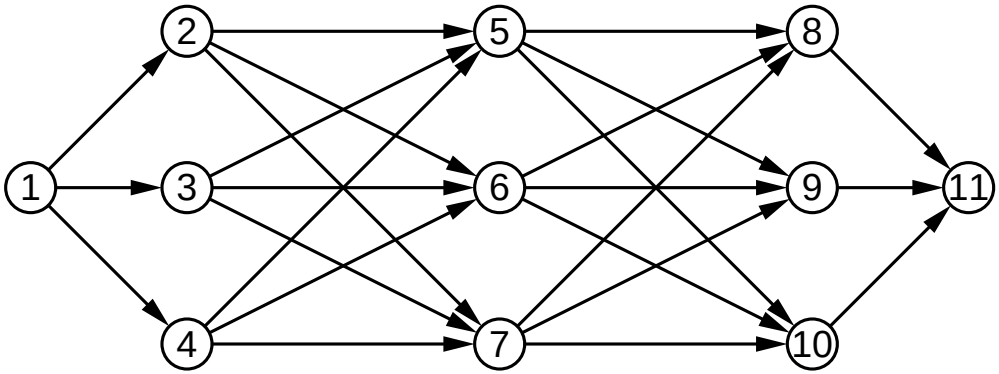
Maxflöde: Exempel



Maxflöde uppnått. Minsnittet går mellan märkta och omärkta noder.
Maxflöde: 5.

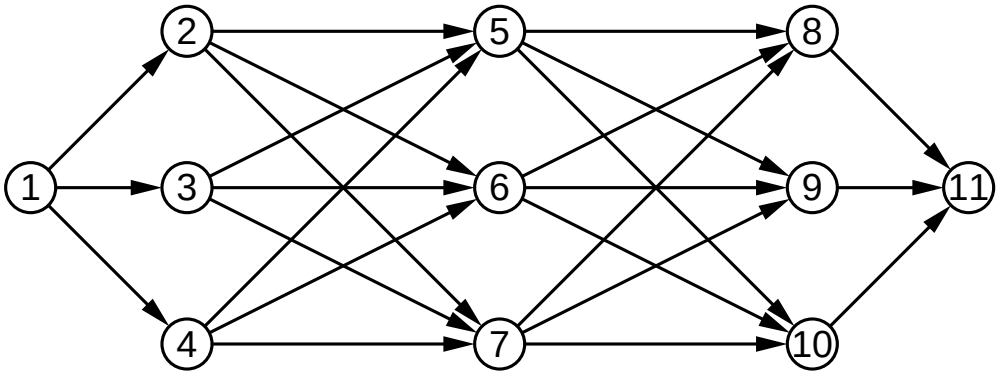
Vägar: Dynamisk programmering

Billigaste väg-problem i acyklisk nivåindelad graf. Tid!



Vägar: Dynamisk programmering

Billigaste väg-problem i acyklisk nivåindelad graf. Tid!



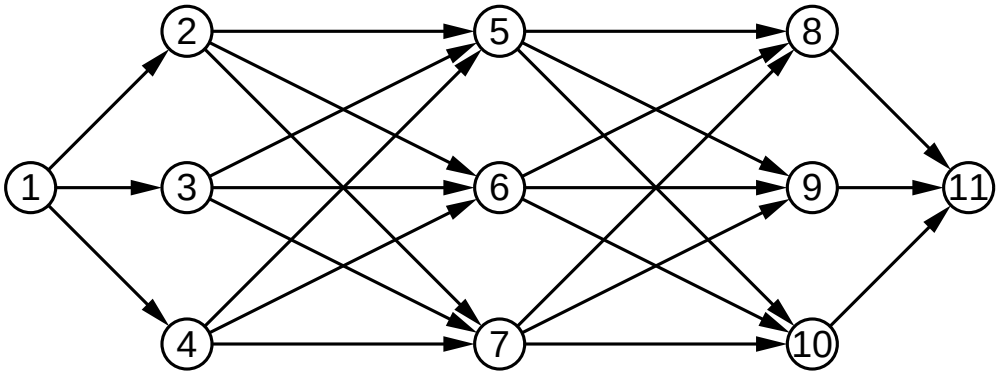
Nodmärkningsmetod (Bellmans ekvationer): Ta en nivå, S_k, i taget:

- $y_1 = 0$.
- För $k = 1, \ldots, N$:

  För varje $j \in S_k$: Sätt $y_j = \min_{i \in S_{k-1}} (c_{ij} + y_i)$.

Vägar: Dynamisk programmering

Billigaste väg-problem i acyklisk nivåindelad graf. Tid!



Nodmärkningsmetod (Bellmans ekvationer): Ta en nivå, S_k, i taget:

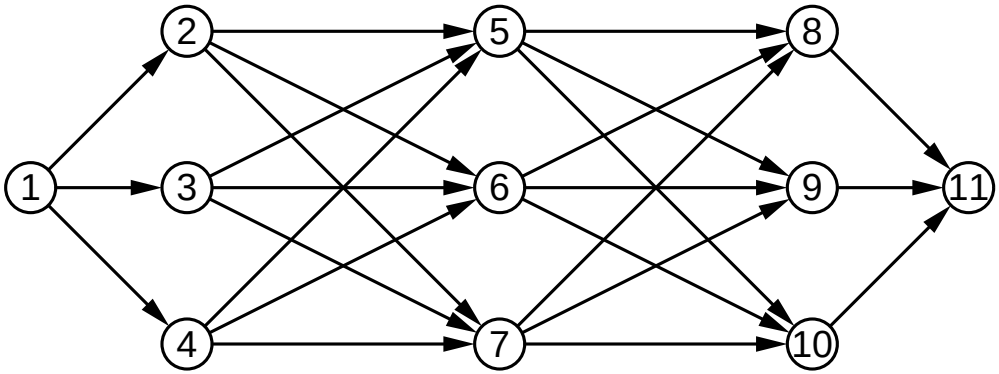
- $y_1 = 0$.
- För $k = 1, \ldots, N$:

  För varje $j \in S_k$: Sätt $y_j = \min_{i \in S_{k-1}} (c_{ij} + y_i)$.

Ordningen inom en nivå oviktig.

Vägar: Dynamisk programmering

Billigaste väg-problem i acyklisk nivåindelad graf. Tid!



Nodmärkningsmetod (Bellmans ekvationer): Ta en nivå, S_k, i taget:

- $y_1 = 0$.
- För $k = 1, \ldots, N$:

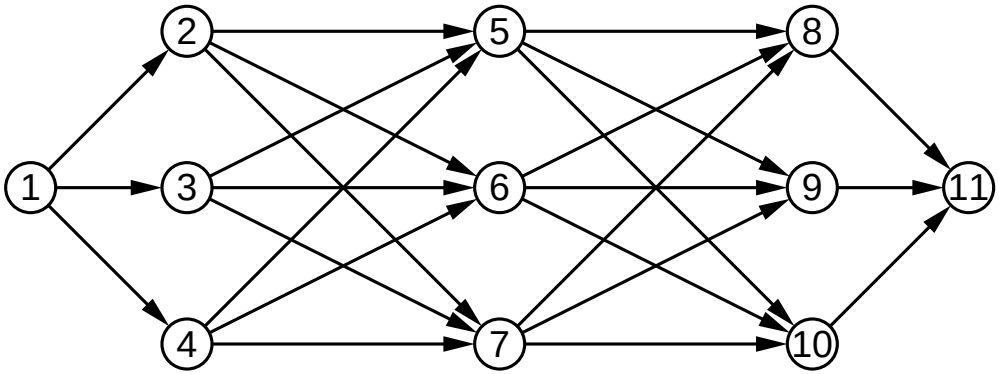
  För varje $j \in S_k$: Sätt $y_j = \min_{i \in S_{k-1}} (c_{ij} + y_i)$.

Ordningen inom en nivå oviktig.

Vägen kommer att passera *en* av noderna i varje nivå. Vet ej vilken.

Vägar: Dynamisk programmering

Billigaste väg-problem i acyklisk nivåindelad graf. Tid!



Nodmärkningsmetod (Bellmans ekvationer): Ta en nivå, S_k, i taget:

- $y_1 = 0$.
- För $k = 1, \ldots, N$:

  För varje $j \in S_k$: Sätt $y_j = \min_{i \in S_{k-1}} (c_{ij} + y_i)$.

Ordningen inom en nivå oviktig.

Vägen kommer att passera *en* av noderna i varje nivå. Vet ej vilken.

Möjliga målfunktioner: $\min \sum$, $\max \sum$, $\min \prod$, $\max \prod$, $\min \max$, $\max \min$.

Dynamisk programmering: Lagerhållningsproblem

Kan lösa allt som "är" ett väg-problem i en acyklisk nivåindelad graf.

Dynamisk programmering: Lagerhållningsproblem

Kan lösa allt som "är" ett väg-problem i en acyklisk nivåindelad graf.

Exempel: **Lagerhållningsproblem**

Dynamisk programmering: Lagerhållningsproblem

Kan lösa allt som "är" ett väg-problem i en acyklisk nivåindelad graf.

Exempel: **Lagerhållningsproblem**

Nod: s_k = antal enheter i lager efter period k.

Dynamisk programmering: Lagerhållningsproblem

Kan lösa allt som "är" ett väg-problem i en acyklisk nivåindelad graf.

Exempel: **Lagerhållningsproblem**

Nod: s_k = antal enheter i lager efter period k.

Hur komma dit: x_k = antal enheter som köps/produceras/säljs i period k.

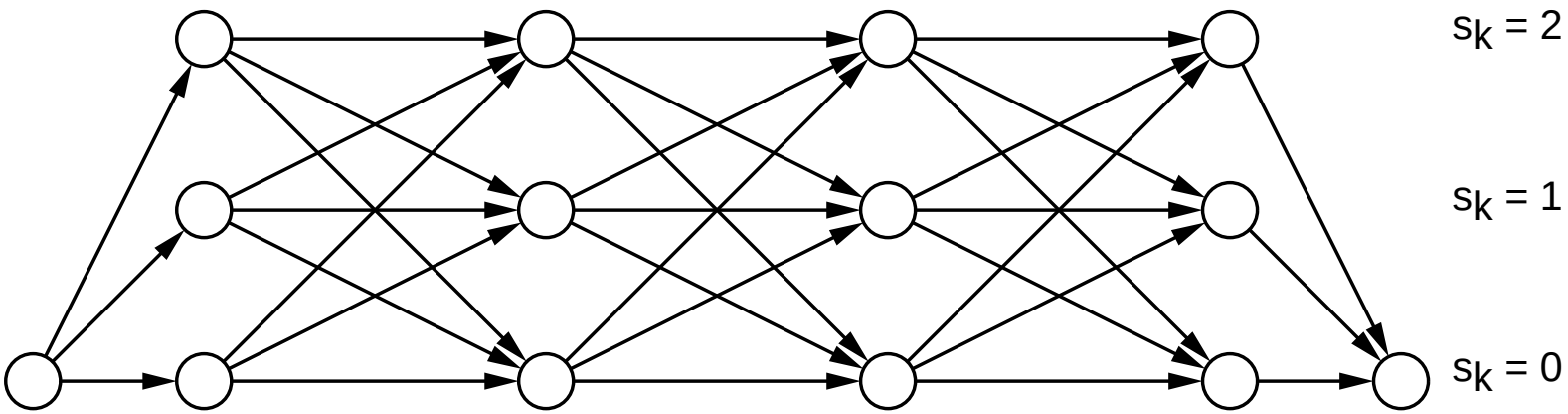
Dynamisk programmering: Lagerhållningsproblem

Kan lösa allt som "är" ett väg-problem i en acyklisk nivåindelad graf.

Exempel: **Lagerhållningsproblem**

Nod: s_k = antal enheter i lager efter period k.

Hur komma dit: x_k = antal enheter som köps/produceras/säljs i period k.



Dynamisk programmering: Kappsäcksproblem

$$\min \sum_j c_j x_j \quad \text{då} \quad \sum_j a_j x_j \leq b, \quad 0 \leq x_j \leq u_j \text{ och heltal, för alla } j$$

Dynamisk programmering: Kappsäcksproblem

$$\min \sum_j c_j x_j \quad \text{då} \quad \sum_j a_j x_j \leq b, \quad 0 \leq x_j \leq u_j \text{ och heltal, för alla } j$$

Varje variabel ses som en nivå.

Dynamisk programmering: Kappsäcksproblem

$$\min \sum_j c_j x_j \quad \text{då} \quad \sum_j a_j x_j \leq b, \quad 0 \leq x_j \leq u_j \text{ och heltal, för alla } j$$

Varje variabel ses som en nivå.

Varje nivå innebär ett ökat utnyttjande av den gemensamma resursen b.

Dynamisk programmering: Kappsäcksproblem

$$\min \sum_j c_j x_j \quad \text{då} \quad \sum_j a_j x_j \leq b, \quad 0 \leq x_j \leq u_j \text{ och heltal, för alla } j$$

Varje variabel ses som en nivå.

Varje nivå innebär ett ökat utnyttjande av den gemensamma resursen b.

Nod: s_k = den del av högerledet b som får användas till de k första variablerna.

Dynamisk programmering: Kappsäcksproblem

$$\min \sum_j c_j x_j \quad \text{då} \quad \sum_j a_j x_j \leq b, \quad 0 \leq x_j \leq u_j \text{ och heltal, för alla } j$$

Varje variabel ses som en nivå.

Varje nivå innebär ett ökat utnyttjande av den gemensamma resursen b.

Nod: s_k = den del av högerledet b som får användas till de k första variablerna.

Koppling mellan nivåerna: $s_{k-1} = s_k - a_k x_k$

Dynamisk programmering: Kappsäcksproblem

Ex: $\max 7x_1 + 2x_2 + 4x_3$

$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,$$

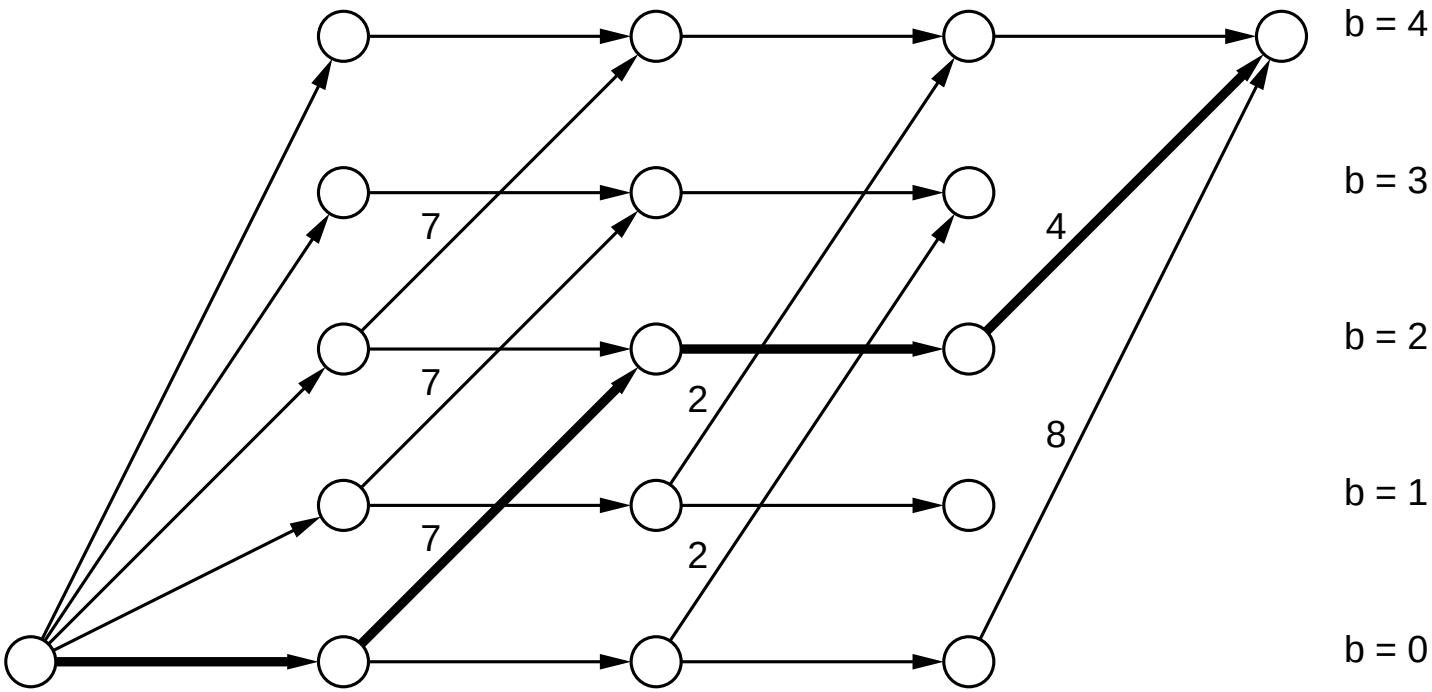
$$x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Dynamisk programmering: Kappsäcksproblem

Ex: $\max 7x_1 + 2x_2 + 4x_3$

då $2x_1 + 3x_2 + 2x_3 \leq 4$,

$x_1 \in \{0, 1\}$, $x_2 \in \{0, 1\}$, $x_3 \in \{0, 1, 2\}$



Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

x_1 \ s_1	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

x_1 \ s_1	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

x_2 \ s_2	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

x_1 \ s_1	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

x_2 \ s_2	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

x_3 \ s_3	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

x_1 \ s_1	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

x_2 \ s_2	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

x_3 \ s_3	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Uppnystning:

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

x_1 \ s_1	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

x_2 \ s_2	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

x_3 \ s_3	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Uppnystning:

$s_3 = 4$,

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

x_1 \ s_1	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

x_2 \ s_2	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

x_3 \ s_3	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Uppnystning:

$s_3 = 4, \quad x_3 = 1,$

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

x_1 \ s_1	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

x_2 \ s_2	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

x_3 \ s_3	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Uppnystning:

$s_3 = 4, \quad x_3 = 1,$

$s_2 = 2,$

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

$x_1 \backslash s_1$	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

$x_2 \backslash s_2$	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

$x_3 \backslash s_3$	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Uppnystning:

$s_3 = 4, \quad x_3 = 1,$

$s_2 = 2, \quad x_2 = 0,$

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

x_1 \ s_1	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

x_2 \ s_2	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

x_3 \ s_3	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Uppnystning:

$s_3 = 4, \quad x_3 = 1,$
$s_2 = 2, \quad x_2 = 0,$
$s_1 = 2,$

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

$x_1 \backslash s_1$	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

$x_2 \backslash s_2$	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

$x_3 \backslash s_3$	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Uppnystning:

$s_3 = 4, \quad x_3 = 1,$
$s_2 = 2, \quad x_2 = 0,$
$s_1 = 2, \quad x_1 = 1,$

Dynamisk programmering: Beräkningar

$$\max 7x_1 + 2x_2 + 4x_3$$
$$\text{då } 2x_1 + 3x_2 + 2x_3 \leq 4,\ x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1,2\}$$

Kan göra beräkningarna i en tabell p.g.a. strukturen hos nätverket.

$x_1 \backslash s_1$	0	1	2	3	4
0	0	0	0	0	0
1	-	-	7	7	7
$f_1(s_1)$	0	0	7	7	7
$\hat{x}_1(s_1)$	0	0	1	1	1

$x_2 \backslash s_2$	0	1	2	3	4
0	0	0	7	7	7
1	-	-	-	2	2
$f_2(s_2)$	0	0	7	7	7
$\hat{x}_2(s_2)$	0	0	0	0	0

$x_3 \backslash s_3$	0	1	2	3	4
0	0	0	7	7	7
1	-	-	4	4	11
2	-	-	-	-	8
$f_3(s_3)$	0	0	7	7	11
$\hat{x}_3(s_3)$	0	0	0	0	1

Uppnystning:

$s_3 = 4, \quad x_3 = 1,$
$s_2 = 2, \quad x_2 = 0,$
$s_1 = 2, \quad x_1 = 1,$
$z = 11.$