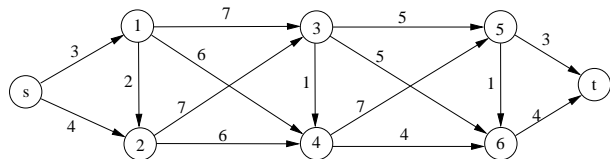


## Vägar: Billigaste väg

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



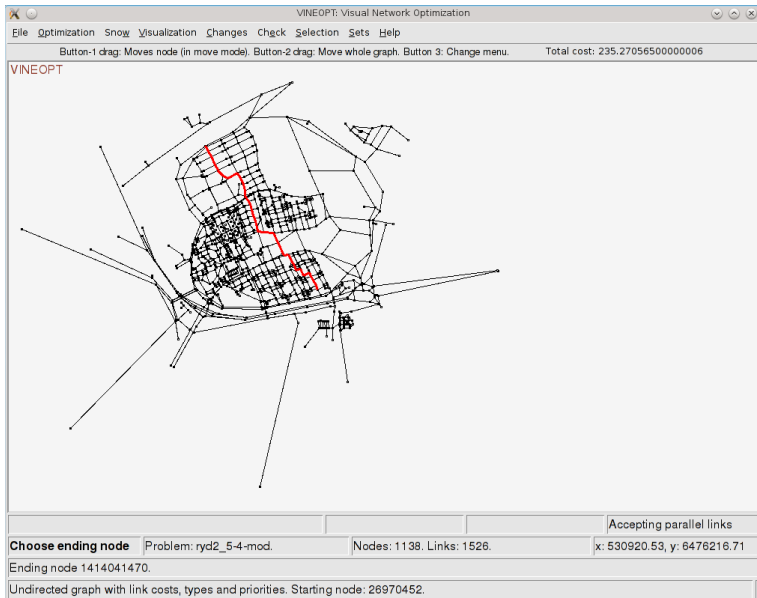
Indata: Riktad graf med bågkostnader  $c$ , start/slutnod  $s$ ,  $t$ .

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: Matematisk modell

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

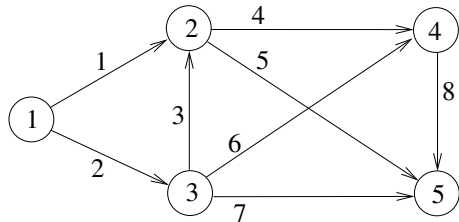
$$\min \sum_{(i,j) \in B} c_{ij} x_{ij}$$

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

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

Nodjämviktsvillkor: (in - ut)

# Exempel



$$-x_{12} - x_{13} = -1 \quad (1)$$

$$x_{12} + x_{32} - x_{24} - x_{25} = 0 \quad (2)$$

$$x_{13} - x_{32} - x_{34} - x_{35} = 0 \quad (3)$$

$$x_{24} + x_{34} - x_{45} = 0 \quad (4)$$

$$x_{25} + x_{35} + x_{45} = 1 \quad (5)$$

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.

## 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:

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

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

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

## 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  $0 = 0$ .

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

$y_j$ : dualvariabel för nod  $i$ .

LP-dualen:

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

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  $y_s = 0$ )

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

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

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

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

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 avsökta.

Om en avsökta nod får lägre nodpris, blir den oavsökta.

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.
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.

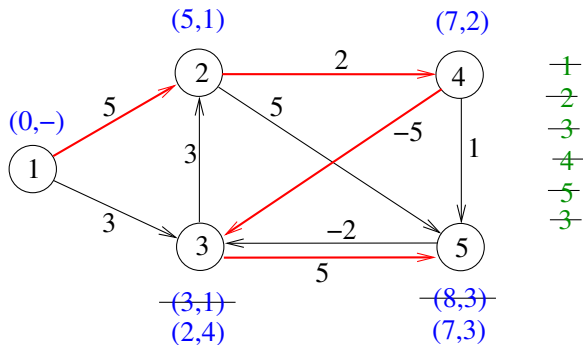
Billigaste vägen nystas upp bakifrån mha  $p_j$ .

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.

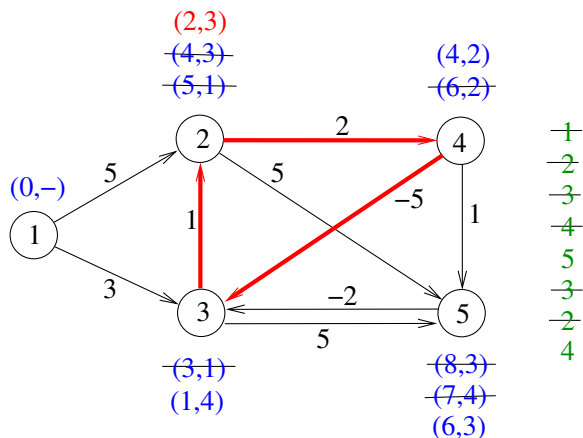
# Fords metod för billigaste väg: Exempel



Alla noder avsökta. Finn väg baklänges.

Billigaste väg: 1 - 2 - 4 - 3 - 5. Kostnad: 7.

## 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

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.

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  $j \notin A$ .

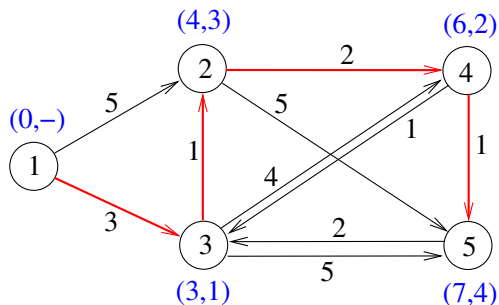
4. Gå till 1.

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



Nysta upp vägen baklänges.

Billigaste väg: 1 - 3 - 2 - 4 - 5. Kostnad: 7.

Nodpriserna ger optimal duallösning.

# 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  $j = 1, \dots, n$ .

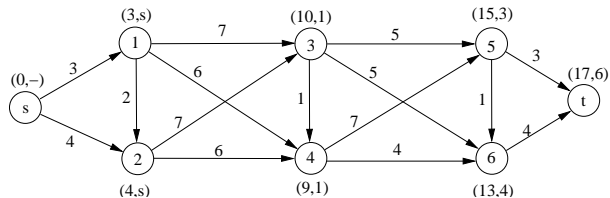
(Bellmans ekvationer. Används för dynamisk programmering.)

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



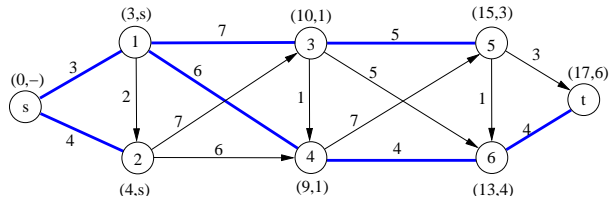
# Billigaste väg

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

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

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

Kan användas för att göra om ett TSPr till ett  $\Delta$ TSP(r).

# 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.

## Slutsats

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

Obs: Inga andra bivillkor får finnas.

## Minkostnadsflödesproblemet

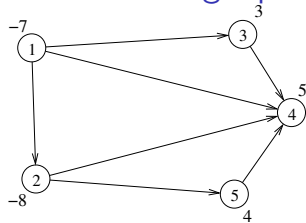
Skicka efterfrågade mängder så billigt som möjligt.

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

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

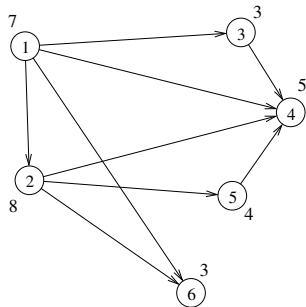
$$l_{ij} \leq x_{ij} \leq u_{ij} \text{ för alla } (i,j) \in B$$

## Nätverksformuleringstips

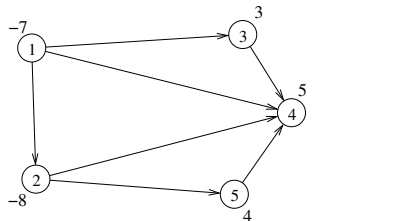


Mer tillgång än efterfrågan: Tillgång: 15, efterfrågan: 12.

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



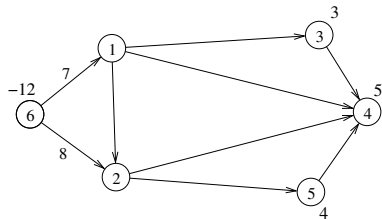
## Nätverksformuleringstips



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

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

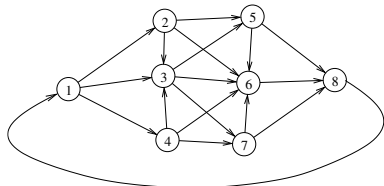
Begränsa flödet i bågar från superkällan till tidigare källor, som nu blir mellannoder.



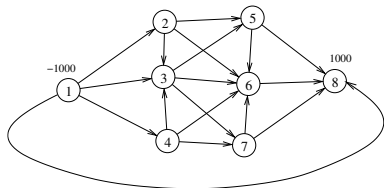
## Nätverksformuleringstips

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

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



Alternativ 2: Inför slackbåge, nod 1 sänka och nod 8 sänka av styrka 1000.





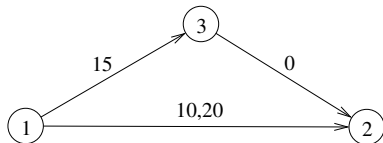
## Nätverksformuleringstips

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, 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  $(i, j)$ .

$$\max \quad f$$

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




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

$f$  fri

# Lösningssmetod 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.

	$x_{ij} = 0$ :	Framåt (öka).	
Tillåtna riktningar:	$0 < x_{ij} < u_{ij}$ :	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  $f^*$ .

## Sats

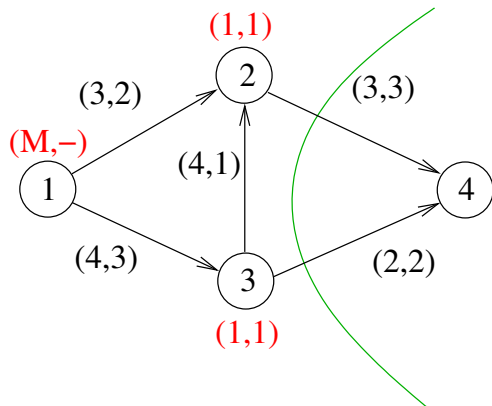
Kapaciteten hos ett *minsnitt* är lika med den maximala flödesstyrkan,  $f^*$ .

## 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).

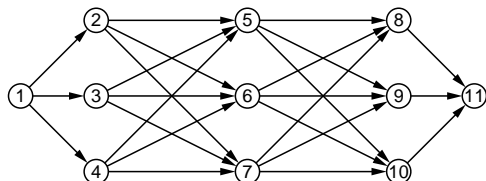
# Maxflöde: Exempel



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

# 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  $k = 1, \dots, N$ :  
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:  $\min \sum$ ,  $\max \sum$ ,  $\min \prod$ ,  $\max \prod$ ,  $\min \max$ ,  $\max \min$ .

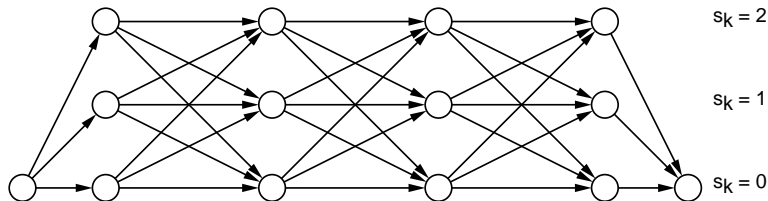
# Dynamisk programmering: Lagerhållningsproblem

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

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

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

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

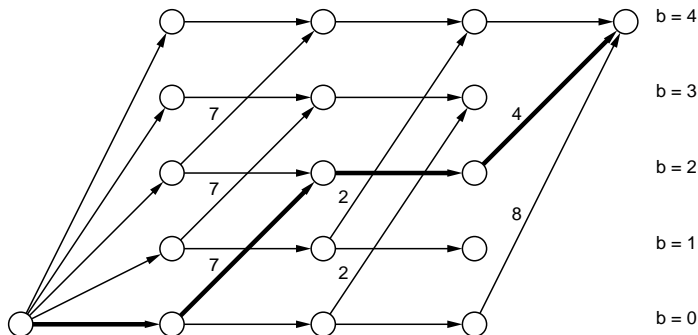


# Dynamisk programmering: Kappsäcksproblem

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

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.

$x_1 \setminus s_1$	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

$x_3 \setminus s_3$	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

$x_2 \setminus s_2$	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

Uppnystning:

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

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

$$s_1 = 2, \quad x_1 = 1,$$

$$z = 11.$$