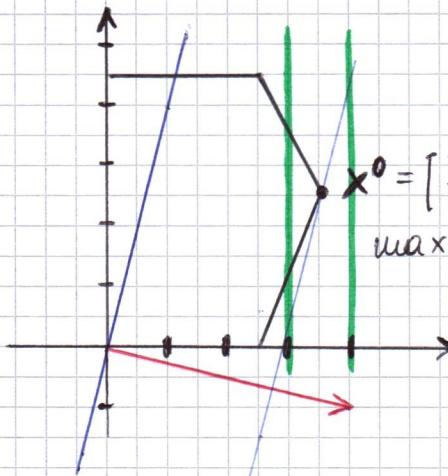


$$\begin{array}{ll} \max & 4x_1 - x_2 \\ & 4x_1 + 2x_2 \leq 19 \\ & 10x_1 - 4x_2 \leq 25 \\ & x_2 \leq 5/2 \\ & x_1, x_2 \geq 0, \in \mathbb{Z} \end{array}$$

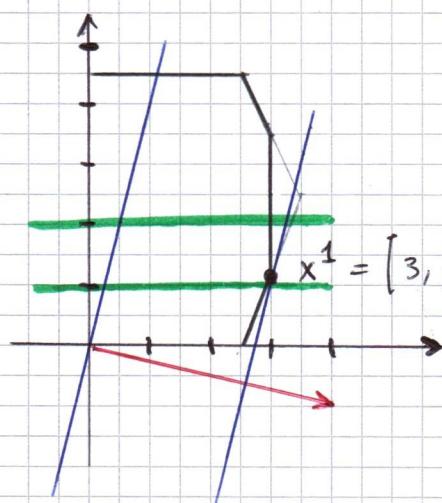


Siccome x^0 non è intero si esegue il branch in x_1 : $x_1 \leq 3, x_1 \geq 4$
($P_1: x \leq 3, P_2: x \geq 4$ non ammissibile)

$$UB_0 = \lfloor 11,5 \rfloor = 11$$

$$x^0 = \left[\frac{7}{2}, \frac{5}{2} \right]$$

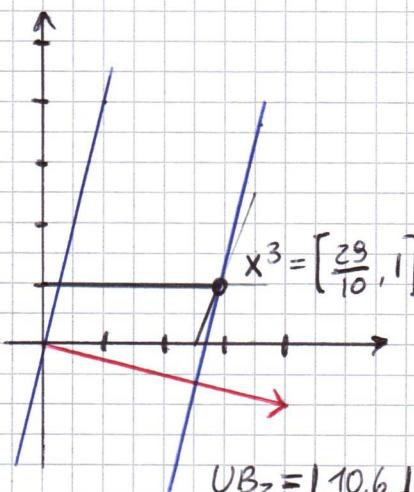
$$\max = 11,5$$



x^1 ancora non è intero, si esegue il branch in x_2 : $x_2 \leq 1, x_2 \geq 2$

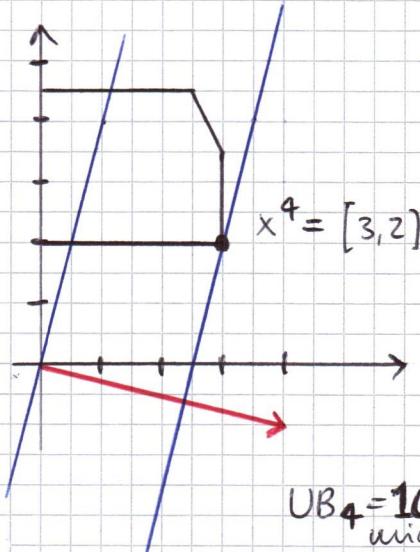
$$UB_1 = \lfloor 10,75 \rfloor = 10$$

$$x^1 = \left[3, \frac{5}{4} \right]$$

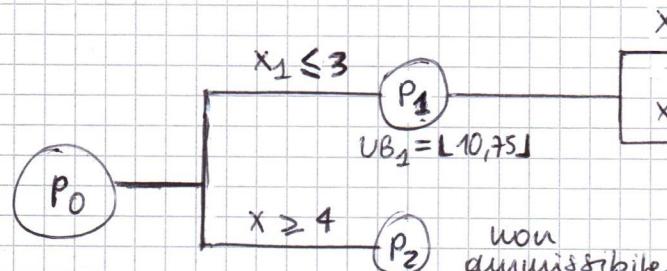


$$UB_3 = \lfloor 10,6 \rfloor$$

$UB_i \leq$ miglior soluzione
intera \Rightarrow si chiude



$UB_4 = 10$ migliore soluzione
intera



$$UB_3 = \lfloor 10,6 \rfloor \leq z^* \text{ si chiude}$$

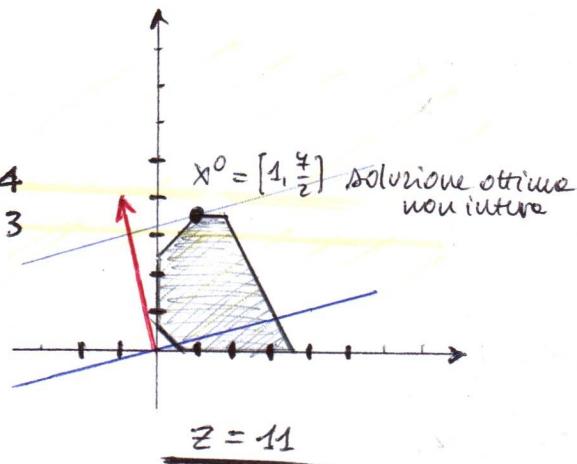
$UB_4 = 10 = z^*$: Soluzione
intera,
no chiuso

B & B

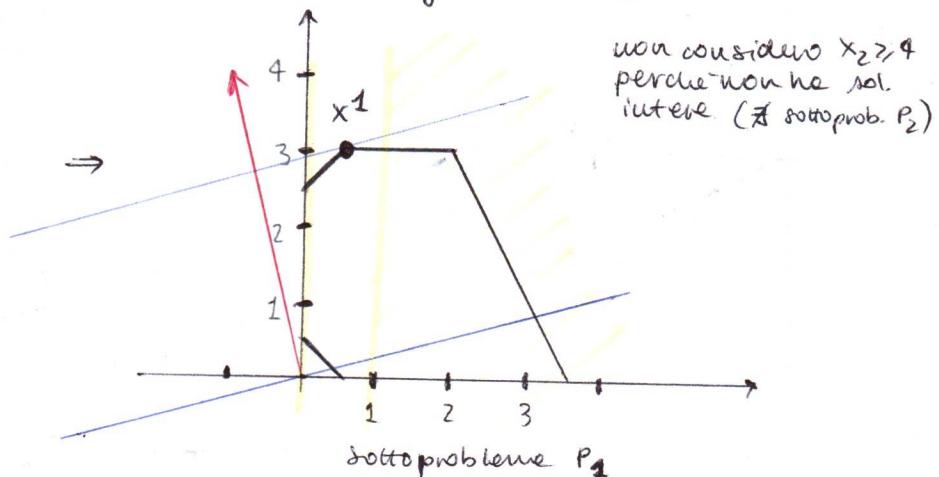
Branch & Bound
(metodo di
enumerazione)

PL I

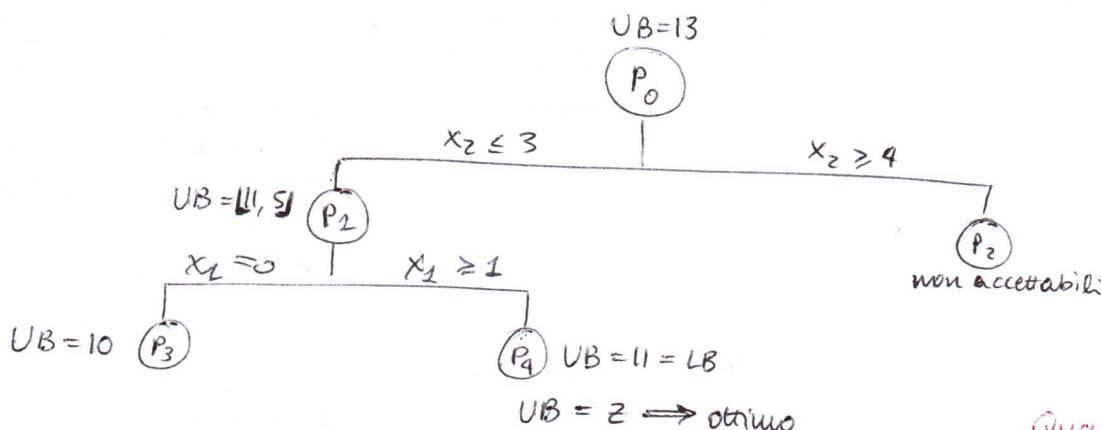
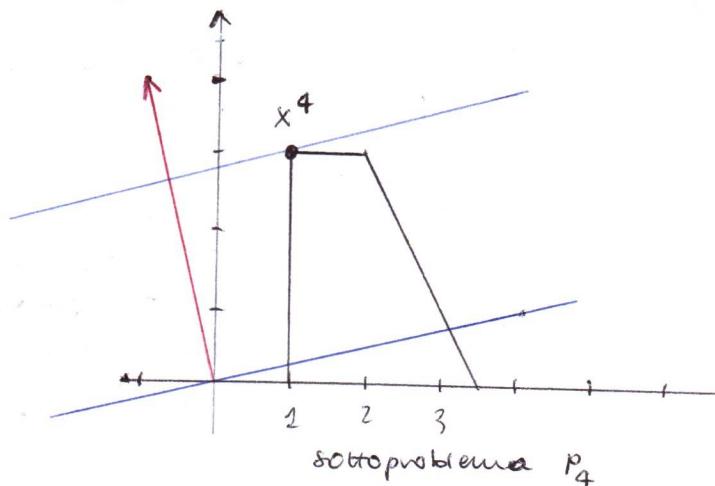
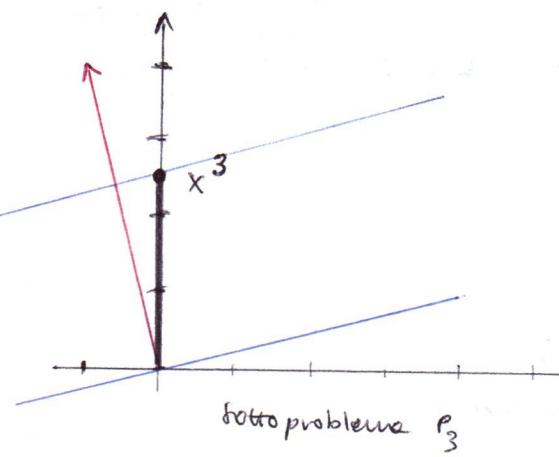
$$\begin{aligned} \text{max } & -x_1 + 4x_2 \\ \text{s.t. } & x_1 + x_2 \geq 1 \\ & x_1 - x_2 \geq -\frac{5}{2} \\ & 2x_1 + x_2 \leq 7 \\ & x_2 \leq \frac{7}{2} \\ & x_1, x_2 \geq 0 \in \mathbb{Z} \end{aligned}$$



I branching: divido $x_2 \geq 4$, $x_2 \leq 3$



II branching: divido $x_1 \leq 0$, $x_1 \geq 1$



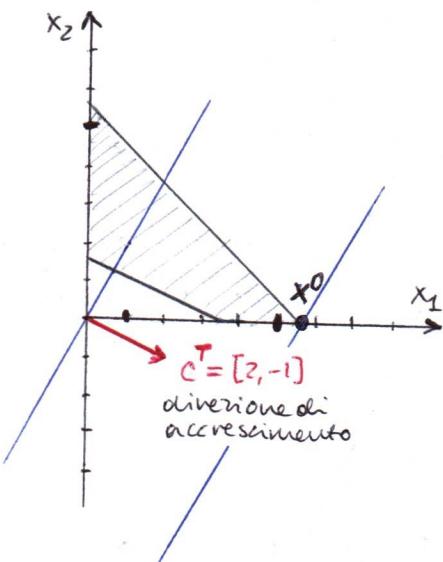
Quando si fanno i tagli
deve sempre rimanere fuori
la soluzione non intera!

$$\begin{aligned}
 & \max 2x_1 - x_2 \\
 & 6x_1 + 14x_2 \geq 21 \\
 & 2x_1 + 2x_2 \leq 11 \\
 & x_1, x_2 \geq 0, \text{ intere}
 \end{aligned}$$

GOMORY

metodo dei
piani di taglio

PLI



La soluzione ottima (ottenuta graficamente) è $x^0 = \left(\frac{11}{2}, 0\right)$

$$\text{Formule std. } \min -2x_1 + x_2$$

$$6x_1 + 14x_2 - x_3 = 21$$

$$2x_1 + 2x_2 + x_4 = 11$$

$$x_1, x_2, x_3, x_4 \geq 0$$

$$\text{Base } \{x_1, x_3\} : A_B = \begin{bmatrix} 6 & -1 \\ 2 & 0 \end{bmatrix}, A_B^{-1} = \begin{bmatrix} 0 & \frac{1}{2} \\ -1 & 3 \end{bmatrix}$$

$$x_B + A_B^{-1} A_N x_N = A_B^{-1} b$$

$$\begin{bmatrix} x_1 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 & \frac{1}{2} \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} 11/2 \\ 12 \end{bmatrix}$$

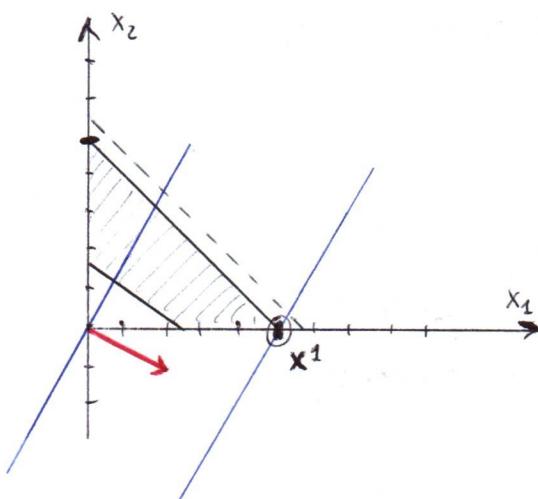
$$\Rightarrow \begin{aligned} x_1 + x_2 + \frac{1}{2}x_4 &= 11/2 \\ x_3 + 2x_2 + 3x_4 &= 12 \end{aligned}$$

$$x_h + \sum_{j \in N} L a_{tj} \lfloor x_j \rfloor \leq L b_t \lfloor$$

prendi la riga con
il b negativo e
avanzando tutto
per difetto:

$$\Rightarrow x_1 + x_2 \leq 5 \quad \Rightarrow$$

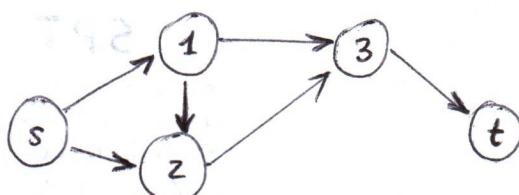
$$\begin{aligned}
 & \min -2x_1 + x_2 \\
 & 6x_1 + 14x_2 - x_3 = 21 \\
 & 2x_1 + 2x_2 + x_4 = 11 \\
 & x_1 + x_2 + x_5 = 5 \\
 & x_1, x_2, x_3, x_4, x_5 \geq 0
 \end{aligned}$$



$x^1 = [1, 0]$: soluzione ottima del
riconcavo continuo

ricorre e' intera \Rightarrow la soluzione
e' ottima

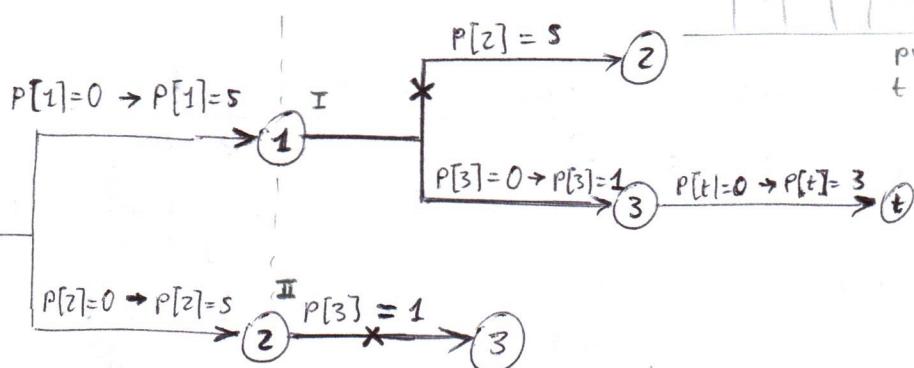
N.B. se in * ci fosse stato anche x_3
o x_4 allora si consideravano le
altre variazioni per ricavare
il vincolo * senza x_3 e x_4



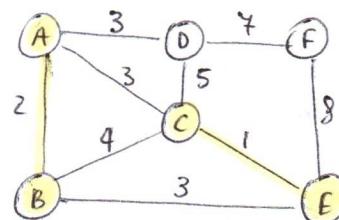
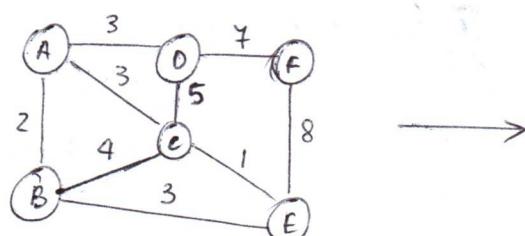
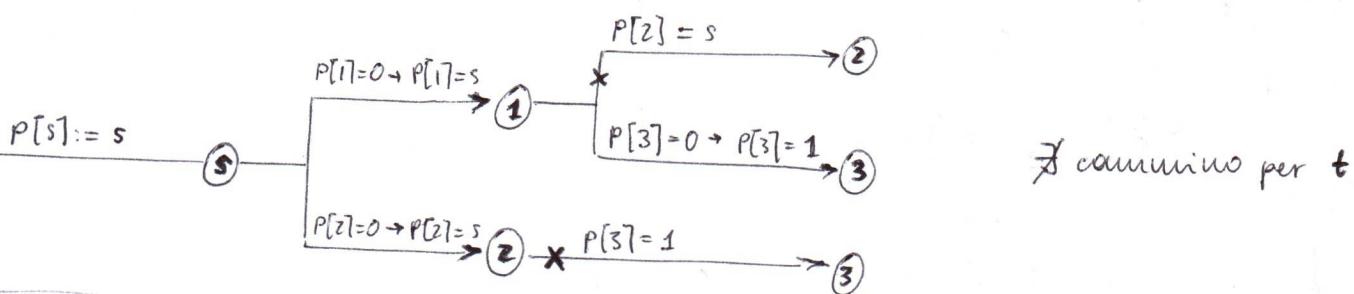
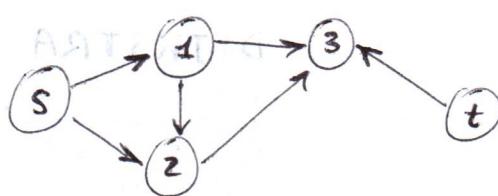
now in

	S	1	2	3	t
S	x	x			
1		x	x		
2			x		
3				x	
t					x

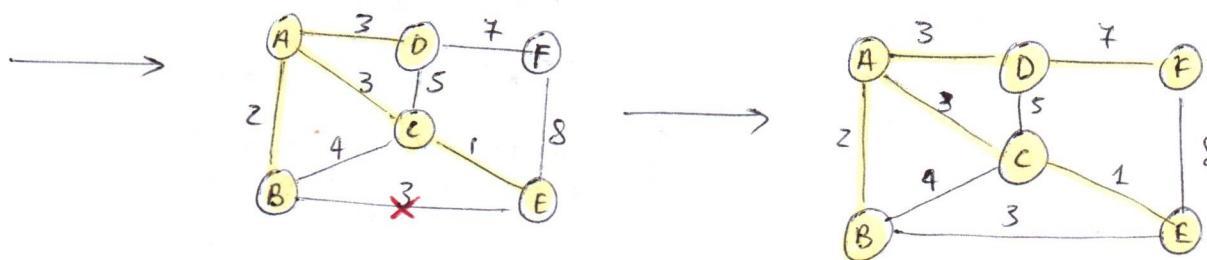
unicollege con



$t \leftarrow 3 \leftarrow 1 \leftarrow s$



KRUSKAL
albero di copertura
di costo minimo



SPT

albero del cammino minimo
GRAFO ACICLICO
($c_{ij} \geq 0$)

- non torna sui nodi già visitati
- non aggiorna le etichette

Percorro i nodi ordinatamente (ordinamento topologico!)

$$\textcircled{1}: P[1] = 1 \\ d_1 = 0$$

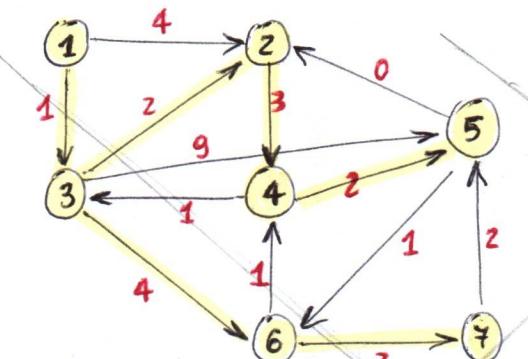
$$1 \xrightarrow{5} 2 : P[2] = 1 \\ d_2 = 5$$

$$2 \xrightarrow{6} 3 : P[3] = 2 \\ d_3 = 6$$

$$2 \xrightarrow{d_2+1} 4 : P[4] = 2 \\ d_4 = 6$$

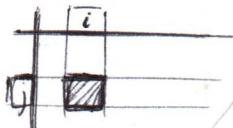
$$2 \xrightarrow{7} 5 : P[5] = 4 \\ d_5 = 7$$

$$4 \xrightarrow{8} 6 : P[6] = 4 \\ d_6 = 8$$

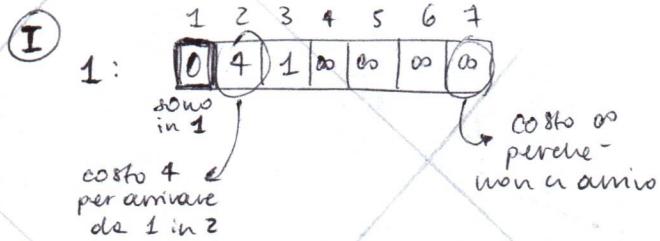


	1	2	3	4	5	6	7
1	0	4	1	00	00	00	00
3	00	3	1	00	10	5	00
2	00	3	00	6	00	00	00
4	00	00	7	6	8	00	00
5	00	8	00	00	8	9	00
6	00	00	00	6	00	5	7
7	00	00	00	00	9	00	7

Di riassegnare colonna evidenziando (la prima volta che si presenta) la cifra più bassa:



$$P[i] = j$$



II: Guardando la riga di 1 prendo il non visitato con costo più basso: 3

1	2	3	4	5	6	7
00	3	1	00	10	5	00

III: Guardando la riga di 3 prendo il non visitato con costo più basso: 2 :

1	2	3	4	5	6	7
00	3	00	6	00	00	00

scrivo il costo inferiore tra quelli fin'ora

[...] ottengo le tabelle piena

DJIKSTRA

albero del cammino minimo
GRAFO ACICLICO
($c_{ij} \geq 0$)

- non torna sui nodi già visitati
- aggiorna le etichette

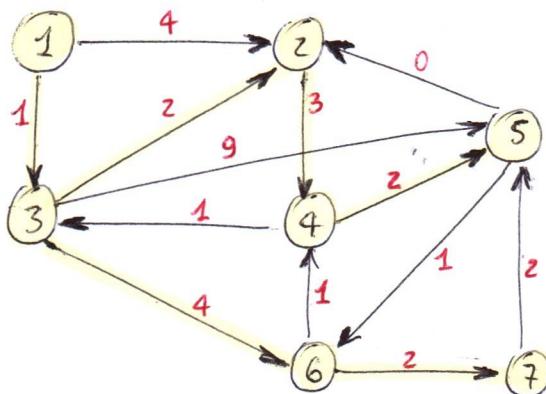
D1JKSTRA

albero del cammino minimo

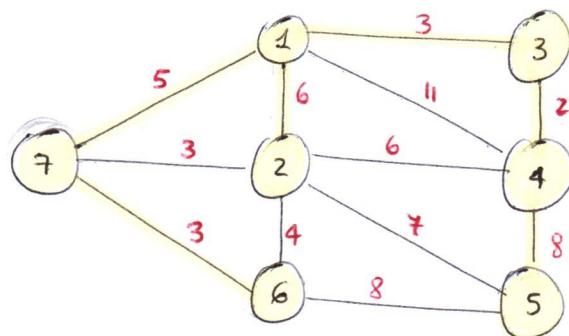
GRAFO CIClico

($c_{ij} \geq 0$)

- uscita da un nodo
- modifica le etichette



$d[1]$	$p[1]$	0, 1						
2		00, 1	4, 1	3, 3				
3		00, 1	3, 1					
4		00, 1			6, 2			
5		00, 1		10, 3				
6		00, 1		5, 3			8, 4	
7		00, 1			7, 6			
Q		1	2, 3	2, 5, 6	5, 6, 4	5, 4, 7	5, 7	5
i		1	3	2	6	4	7	5

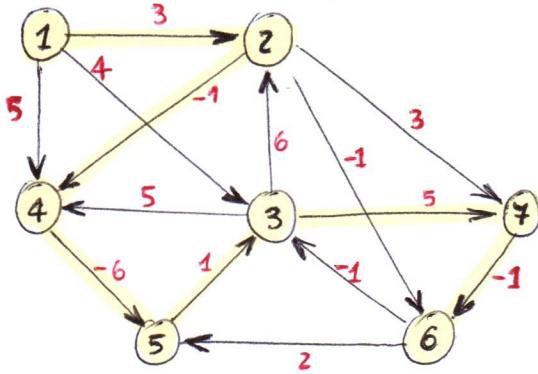


$d[1], p[1]$	0, 1							
2	00, 1	6, 1						
3	00, 1	3, 1						
4	00, 1	11, 1	5, 3					
5	00, 1			13, 4				
6	00, 1				8, 7			
7	00, 1	5, 1						
Q		1	2, 3, 4, 7	2, 4, 7	2, 7, 5	2, 5, 6	5, 6	5
i		1	3	4	7	2	6	5

BELLMAN FORD

albero del cammino minimo
GRAFO CICLICO
 $c_{ij} \geq 0$

- forma svincola'
- aggiornamento etichette



$$\begin{aligned} d[j] &> d[i] + c_{ij} \\ \Rightarrow d[j] &= d[i] + c_{ij} \\ p[j] &= i \\ k[j] &= k[i] + 1 \text{ se } j \notin Q \end{aligned}$$

1 : $d[1], p[1], k[1]$	0, 1, 1										
2	00	3, 1, 1									
3	00	4, 1, 1									
4	00	5, 1, 1	2, 2, 1								
5	00										
6	00		2, 2, 2								
7	00		6, 2, 1								
Q	1	2, 3, 4	3, 4, 6, 7	4, 6, 7	6, 7, 5	7, 5, 3	5, 3	3	7	6	
i	1	2	3	4	6	7	5	3	7	6	
	I		II	III	IV	V	VI	VII	VIII	IX	X

I) $i=1$:
 $(1,2) \quad 00 > 0 + 3 = 3 \Rightarrow d[2] = 3 \quad p[2] = 1 \quad k[2] = 1$
 $(1,3) \quad 00 > 0 + 4 = 4 \Rightarrow d[3] = 4 \quad p[3] = 1 \quad k[3] = 1$
 $(1,4) \quad 00 > 0 + 5 = 5 \Rightarrow d[4] = 5 \quad p[4] = 1 \quad k[4] = 1$

II) $i=2$:
 $(2,4) \quad 5 > 3 - 1 = 2 \Rightarrow d[4] = 2 \quad p[4] = 2 \quad k[4] = 1$
 $(2,7) \quad 00 > 3 + 3 = 6 \Rightarrow d[7] = 6 \quad p[7] = 2 \quad k[6] = 1$
 $(2,6) \quad 00 > 3 - 1 = 2 \Rightarrow d[6] = 2 \quad p[6] = 2 \quad k[7] = 1$

III) $i=3$
 $(3,2) \quad 3 > 4 + 6 = 10 \quad \text{NO}$
 $(3,4) \quad 2 > 4 + 5 = 9 \quad \text{NO}$
 $(3,7) \quad 6 > 4 + 5 = 9 \quad \text{NO}$ } \Rightarrow non ci sono cambiamenti

IV) $i=4$ $(4,5) \quad 00 > 2 - 6 = -4 \Rightarrow d[5] = -4 \quad p[5] = 4 \quad k[5] = 1$

V) $i=6$ $(6,3) \quad 4 > 2 - 1 = 1 \Rightarrow d[3] = 1 \quad p[3] = 6 \quad k[3] = 2$
 $(6,5) \quad -4 > 2 + 2 = 4$

VI) $i=7$ $(7,6) \quad 2 > 6 - 1 = 5$

VII) $i=5$ $(5,3) \quad 1 > -4 + 1 = -3 \Rightarrow d[3] = -3 \quad p[3] = 5 \quad k[3] = 2$

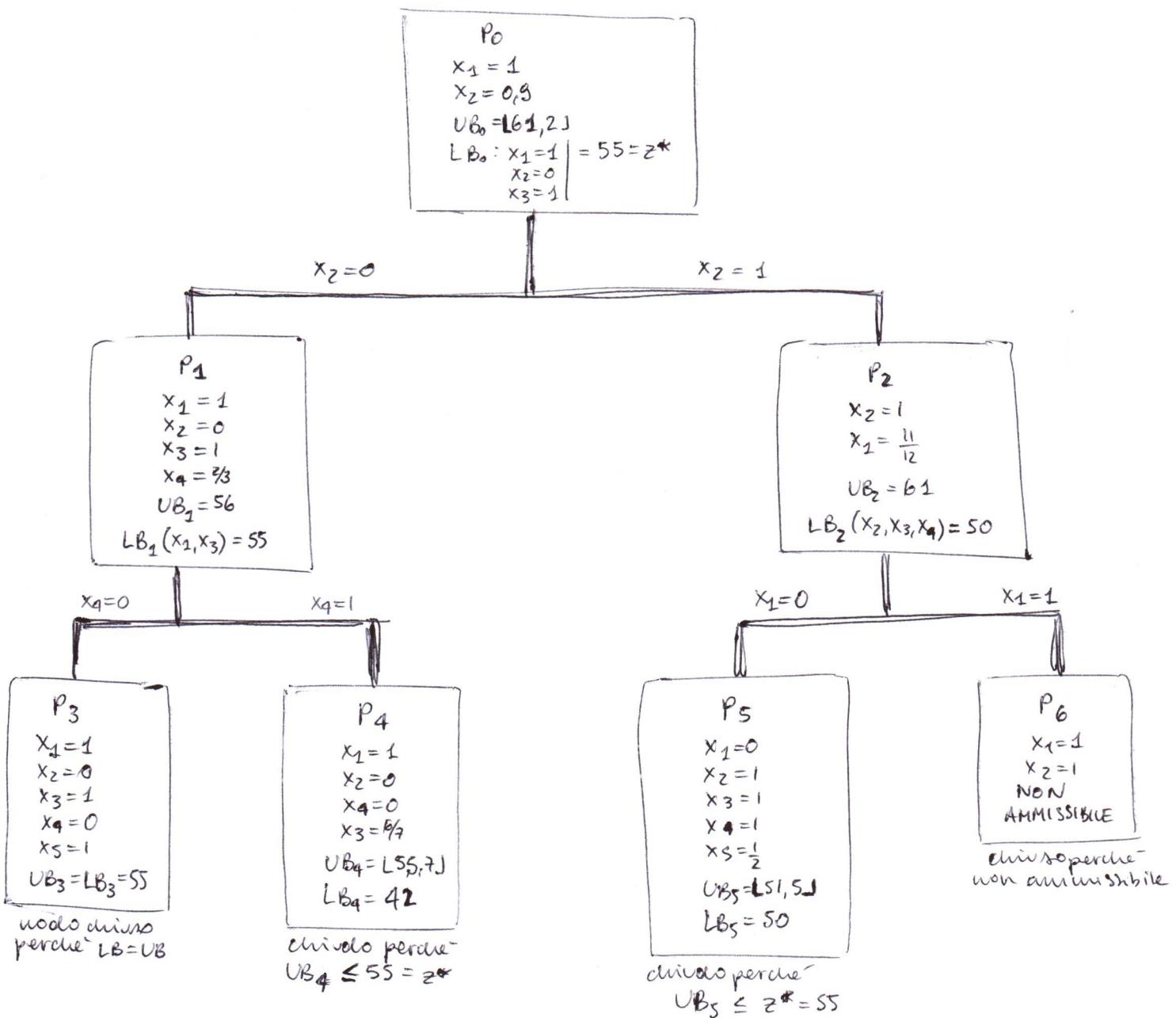
VIII) $i=3$
 $(3,2) \quad 3 > -3 + 6 = 3$
 $(3,4) \quad 2 > -3 + 5 = 2$
 $(3,7) \quad 6 > -3 + 2 = 2 \Rightarrow d[7] = 2 \quad p[7] = 3 \quad k[7] = 2$

IX) $i=7$ $(7,6) \quad 2 > 2 - 1 = 1 \Rightarrow d[6] = 1 \quad p[6] = 7 \quad k[6] = 3$

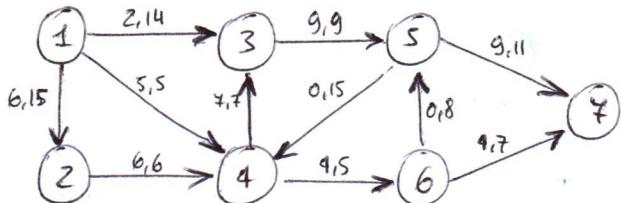
X) $i=6$
 $(6,3) \quad -3 > 1 - 1 = 0$
 $(6,5) \quad -4 > 1 - 2 = -1$ } NO \Rightarrow fine

$$\begin{array}{l} \text{MAX } 36x_1 + 28x_2 + 16x_3 + 6x_4 + 3x_5 \\ 12x_1 + 10x_2 + 4x_3 + 3x_4 + 2x_5 \leq 21 \end{array}$$

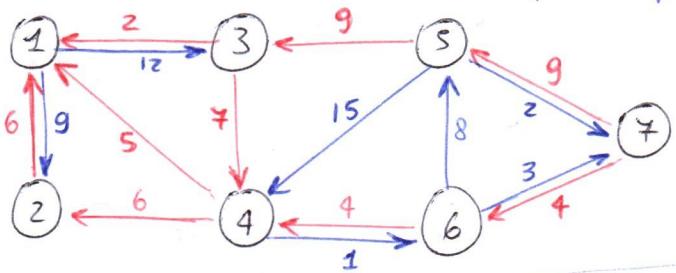
$$\frac{p_i}{w_i} \quad 3 \quad 2,8 \quad 2,28 \quad 2 \quad 1,5$$



**FORD
FULKERSON**
algoritmo dei
cammini aumentati
(flusso massimo)



quanto possono diminuire
quanto possono aumentare



$$\theta = \min \left\{ \min(\bullet), \min(\circ) \right\} = 1$$

$$x_{ij} = \begin{cases} x_{ij} + \theta & (i,j) \in P^+ \\ x_{ij} - \theta & (i,j) \in P^- \\ x_{ij} & \text{altrimenti} \end{cases}$$

venendo modificati
solo quelli del cammino
minimo:

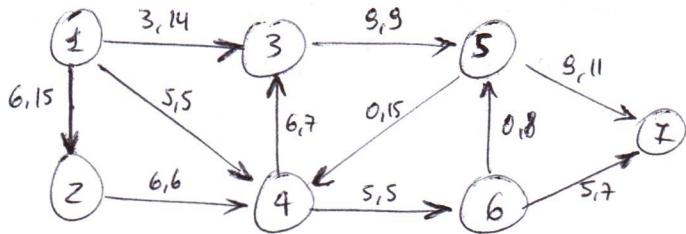
I grafo residuale:

cammino minimo:

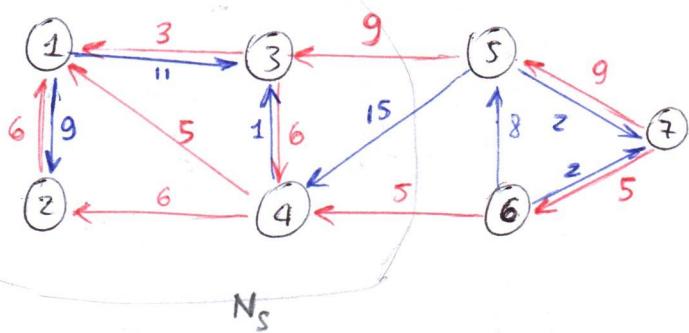
$$P = \{1, 3, 4, 6, 7\}$$

$$P^+ = \{(1,3)^{12}, (4,6)^1, (6,7)^3\}$$

$$P^- = \{(3,4)^7\}$$

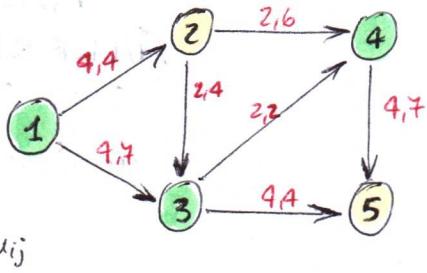


Nuovo flusso:



la somma delle frecce rosse (valori)
e' entranti e' il FLUSSO

$$N_S = \{1, 2, 3, 4\}, N_T = \{5, 6, 7\}$$



x_{ij}, u_{ij}

$$N_S = \{1, 3, 4\}$$

$$N_T = \{2, 5\}$$

$$A^+ = \{(i, j) \in A : i \in N_S, j \in N_T\}$$

$$A^- = \{(i, j) \in A : i \in N_T, j \in N_S\}$$

- Capacità di taglio: $U(N_S, N_T) = \sum_{(i, j) \in A^+} u_{ij}$

mi interessano gli archi:



$$(1, 2) \quad u_{12} = 4$$

$$(3, 5) \quad u_{35} = 4$$

$$(4, 5) \quad u_{45} = 7$$

$$\Rightarrow U(N_S, N_T) = 15$$

- Flusso attraverso un taglio: $X(N_S, N_T) = \sum_{(i, j) \in A^+} x_{ij} - \sum_{(i, j) \in A^-} x_{ij}$

$$= \sum_{\substack{(i, j) \in A^+ \\ \text{in avanti}}} (i \rightarrow j) - \sum_{\substack{(i, j) \in A^- \\ \text{all'indietro}}} (j \rightarrow i)$$

$$X(N_S, N_T) \leq U(N_S, N_T) \longrightarrow$$

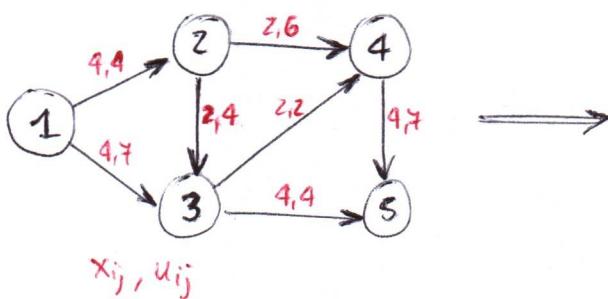
Condizioni di ottimalità:

$$\text{se } X(N_S, N_T) = U(N_S, N_T)$$

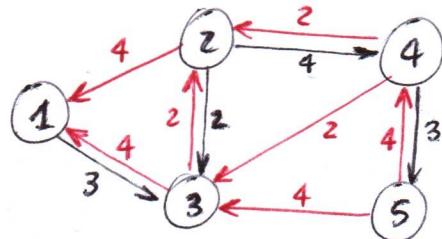
\Rightarrow il flusso è massimo (ottimo)

Grado residuale:
descrivere quali variazioni
sono ammissibili:

le linee nere: "può essere
aumentato di", le rosse:
"può diminuire di"



x_{ij}, u_{ij}



Cammino aumentante:
Se esiste un cammino (detto aumentante)
da s a t sul grafo residuale
 \Rightarrow la soluzione non è ottima

SIMPLEXO

I: (A, b, c) è una base ammissibile
O: soluzione di base ottima
 Parte da una soluzione ammissibile e si muove (nella regione ammissibile) per arrivare all'ottima (costi ridotti non negativi)
 (il problema deve essere scritto in forma standard:
 $\min c^T x : Ax = b, x \geq 0$)
 $z = z_0 + r_N^T x$

- Si calcola:
 valore delle variabili in base: $x_B = A_B^{-1} b$
 valore della funzione obiettivo: $z = c_B^T A_B^{-1} b$
 vettore dei costi ridotti: $r_N^T = c_N^T - c_B^T A_B^{-1} A_N$
- Se i costi ridotti sono tutti positivi (≥ 0) la soluzione è ottima, il simplex termina
- Altrimenti si seleziona una variabile fuori base x_i con costo ridotto negativo
- Se $(A_B^{-1} A_N)_{ji} \leq 0 \quad \forall j \in B$ l'ottimo è illimitato e il simplex termina
- Altrimenti si seleziona la variabile j^* in base:

$$j^* = \arg \min \frac{(A_B^{-1} b)_j}{(A_B^{-1} A_N)_{ji}} \quad j \in B$$

$$(A_B^{-1} A_N)_{ji} > 0$$
- Si aggiorna la base sostituendo x_{j^*} con x_i e si torna al punto 1

SIMPLEXO DUALE

I: soluzione di base super ottima ma non ammissibile
O: soluzione ammissibile ottima si muove verso l'ottimalità duale mantenendo l'ammissibilità duale, ovvero parte da una sol. non ammissibile super ottima primaria e si muove verso l'ammissibilità mantenendola super ottima

- Si calcola: $x_B = A_B^{-1} b, z = c_B^T A_B^{-1} b$.
 Se $\forall q \in B, x_q \geq 0$ la base attuale è ammissibile e ottima (il processo termina)
- Altrimenti selezionare $q \in B: x_q < 0$ (p-esimo elemento della base)
- Se $(A_B^{-1} A_N)_{qi} \geq 0 \quad \forall i \notin B$ il duale è illimitato, il primario non è ammissibile
- Altrimenti entro in base al posto di x_p la variabile x_j :

$$j = \arg \min \left\{ -\frac{r_i}{(A_B^{-1} A_N)_{pi}} : (A_B^{-1} A_N)_{pi} < 0 \right\}, i$$
 torna a 1

ALGORITMO DI VISITA

I: grafo $G = (N, A)$, $s, t \in N$
O: cammino che li collega o taglio che li separa
 L'algoritmo usa una lista Q per contenere i nodi visitati e P per ogni nodo che contiene il nodo che lo precede nel cammino (predecessore)

- (inizializzazione): $P[i] = 0 \quad \forall i \in N, P[s] = s, Q = \{s\}$
- Si estrare un nodo $i \in Q$
- Se $i = t$: cammino trovato, fine dell'algoritmo
- Altrimenti si scorrono tutti gli archi della stessa uscente di i : $\{e(i,j) \in FS(i)\}$:
 se $P[j] = 0 \Rightarrow P[j] = i, Q = Q \cup \{j\}$
- Se $Q = \emptyset$ i nodi non sono连nessi e l'algoritmo termina, altrimenti si torna a 2

KRUSKAL

Albero di copertura di costo minimo
I: grafo $G = (N, E)$, costo di ogni lato
O: albero di costo minimo
 viene costituito aggiungendo un lato alla volta (i nuovi costi). Non si aggiungono lati che creano cicli.

- (iniz) $T = \emptyset$
- Si ordinano i lati per costo non decrescente
- Si seleziona e , il prossimo lato nell'ordine
- Se $T \cup \{e\}$ ha un ciclo si torna a 3
- Altrimenti $T = T \cup \{e\}$
- Se $|T| = n-1$ l'algoritmo termina ($n = |N|$)
- Altrimenti si torna a 3

ENUMERAZIONE

TOPOLOGICA

I: grafo $G = (N, A)$
O: etichette ϕ_i che rappresentano l'ordinamento topologico
 L'algoritmo ha una struttura ricorsiva: si applica a un grafo via via modificato mantenendo il prossimo valore dell'etichetta da assegnare in x

- $G'(N', A') = G(N, A), \quad x = 1$
- Se $|N'| = 0$ allora l'algoritmo termina
- Altrimenti si cerca $i \in N'$: $PN(i) = \emptyset$
 - Se $\exists i \in N': PN(i) = \emptyset$ il grafo non è aciclico e l'algoritmo termina
 - Altrimenti si assegna l'etichetta i , si aggiorna il grafo e il parametro:
 $\phi(i) = x, \quad x = x+1, \quad N' = N' \setminus \{i\}$
 $A' = A' \setminus FS(i)$, si torna a 2

SPT

Albero del cammino minimo

GRAFO ACICLICO ($c_{ij} \geq 0$)

I: grafo $G = (N, A)$, dove i nodi sono ordinati secondo l'ordinamento topologico, costi associati e r

O: cammino minimo da r a ogni altro nodo

Vengono usate due etichette: $d[i]$ che all'altro rappresenta la lunghezza tra i e r e $p[i]$ che rappresenta il predecessore di i nel cammino da r a i

$O(|A|) + \text{costo ordinamento}$

1. (iniz.) $P[i] = r$, $d[i] = \infty \quad \forall i \in N$

2. Si considera i : il prossimo elemento nell'ordinamento topologico

3. Si analizza $FS(i)$:

se $d[i] + c_{ij} < d[j] \Rightarrow d[j] = d[i] + c_{ij}$, $P[j] = i$

4. Se ci sono ancora nodi da esplorare si torna al passo **2**

DIJKSTRA

Albero del cammino minimo

GRAFO CIClico ($c_{ij} \geq 0$)

I: grafo $G = (N, A)$, costi, r

O: cammino minimo da r a ogni altro nodo

Oltre a $d[i]$ e $p[i]$, viene creato Q : un insieme di nodi da visitare

$O(|N|^2)$

1. (iniz.) $P[i] = r$, $d[i] = \infty \quad \forall i \in N$

$P[r] = r$, $d[r] = 0$, $Q = \{r\}$

2. Si seleziona $i \in Q$: $i = \arg \min_{j \in Q} \{d[j]\}$

$Q = Q \setminus \{i\}$

$\forall (i,j) \in FS(i)$ se $d[i] + c_{ij} < d[j]$ allora:

$d[j] = d[i] + c_{ij}$

$P[j] = i$

de $j \notin Q$: $Q = Q \cup \{j\}$

3. Se $Q = \emptyset$ l'algoritmo termina, se no $\rightarrow 2$.

BELLMAN-FORD

Albero del cammino minimo

GRAFO CIClico ($c_{ij} \geq 0$)

I: grafo $G = (N, A)$, costi, r

O: cammino minimo da r a ogni altro nodo o

ciclo negativo

(possono esistere cicli di lunghezza < 0)

Si controllano gli archi per quali si è modificata $d[i]$. Per tenere conto dei nodi di cui bisogna controllare la stessa vicente si usa una lista Q . Se un nodo entra in Q n volte si rileva la presenza di un ciclo negativo (par. $K[i]$)

$O(|N| \cdot |A|)$

1. (iniz.) $P[i] = r$, $d[i] = \infty$, $K[i] = 0 \quad \forall i \in N$

$P[r] = r$, $d[r] = 0$, $Q = \{r\}$, $K[r] = 1$

2. Si seleziona $i \in Q$, $Q = Q \setminus \{i\}$

3. $\forall (i,j) \in FS(i)$:

Se $d[i] + c_{ij} < d[j]$ allora:

• $d[j] = d[i] + c_{ij}$, $P[j] = i$

• se $j \notin Q \Rightarrow Q = Q \cup \{j\}$, $K[j] = K[i] + 1$

• se $K[j] = |N|$ è presente un ciclo negativo e l'algoritmo termina

4. Se $Q = \emptyset$ l'algoritmo termina restituendo i cammini minimi, altrimenti si va a **2**.

FORD - FULKERSON

Algoritmo dei cammini aumentati (flusso massimo)

I: grafo $G = (N, A)$, s, t $\in N$, capacità $U_{ij} \forall (i,j)$

O: Flusso massimo, flussi sugli archi, taglio di capacità minima

$O(\text{pseudopolinomiale})$:

scegliendo sempre il cammino aumentante con il minor numero di archi si garantisce la polinomialità dell'algoritmo

1. Si costruisce il grafo residuale G_R

2. Si cerca un cammino aumentato sul grafo residuale

3. Se il cammino aumentato \exists si aggiorna il Flusso e si torna a **1**

4. Altrimenti l'algoritmo termina

PIANI DI TAGLIO

TAGLI DI GOMORY

I: problema a variabili intere (A, b, c)

O: soluzione ottima intera

Dato una formulazione iniziale si aggiungono iterativamente nuovi vincoli finché non si ha una soluzione ottima intera.

1. Si risolve il problema rilassato P_r ottenendo la soluzione \bar{x}

2. Se \bar{x} è intera la procedura termina

3. Altrimenti:

- 3.1. Si genera un taglio per \bar{x}

- 3.2. Si aggiunge il taglio ai vincoli del problema e si torna al punto 1.

$$\text{tagli di Gomory} \quad x_h + \sum_{j \in N} \lfloor \bar{a}_{tj} \rfloor x_j \leq \lfloor \bar{b}_t \rfloor$$

BRANCH & BOUND

Metodo di enumerazione

I: problema a variabili intere (A, b, c)

O: soluzione ottima intera

Si partiziona la regione ammissibile generando un albero di ricerca. Si valuta ogni regione ottenuta, stimandone il valore dell'ottimo. Si scartano le regioni che non contengono l'ottimo.

1. Se tutti i nodi sono stati esplorati e chiusi l'algoritmo termina e restituisce z^*

2. Altrimenti seleziona un nodo da esplorare, calcola la stima b e controlla i criteri di potatura:

- 2.1. Se il problema non è ammissibile chiude il nodo
- 2.2. Se la stima b è peggiore della miglior soluzione intera trovata z^* si chiude il nodo
- 2.3. Se la stima b corrisponde ad una soluzione ammissibile intera si chiude il nodo e se b è migliore di z^* si aggiorne z^*

3. Se il nodo non è stato chiuso genera figli con il branch e torna al punto 1

- massimo: ottimo $\in [sol\ intera, stima]$ si chiude se $UB_i \leq z^*$
- minimo: ottimo $\in [stima, sol\ intera]$ si chiude se $LB_i \geq z^*$

[LB, UB]

N.B. scendendo la stima peggiora, la sol. intera ammissibile migliora

Zaino binario (B&B)

$$\max \sum p_i x_i$$

$$\sum w_i x_i \leq B$$

$$x_i = \{0, 1\}$$

1. Ordinare gli oggetti per rapporto $\frac{p_i}{w_i}$ non crescente

2. Sia \bar{B} la capacità residua

3. Sia i^* il prossimo oggetto secondo l'ordinamento

- Se $w_{i^*} \leq \bar{B} \Rightarrow x_{i^*} = 1, \bar{B} = \bar{B} - w_{i^*}$

- Se $w_{i^*} > \bar{B}$ e $\bar{B} > 0 \Rightarrow x_{i^*}$ è frazionaria

- Se $\bar{B} = 0 \Rightarrow x_{i^*} = 0 \quad \forall i \geq i^*$

B&B

$$\begin{aligned}
 \max \quad & 2y_1 + y_2 \\
 -2y_1 - y_2 &\leq -1 \\
 y_1 - y_2 &\leq 3 \\
 4y_1 + y_2 &\leq 17 \\
 y_2 &\leq 5 \\
 -y_1 + y_2 &\leq 4
 \end{aligned}$$

$$y_1, y_2 \geq 0$$

Primal	Duale
min	max
$A_i x \geq b$	$y_i \geq 0$
$A_i x \leq b$	$y_i \leq 0$
$A_i x = b$	$y_i \text{ lib.}$
$x_i \geq 0$	$y^T A^i \leq c^i$
$x_i \leq 0$	$y^T A^i \geq c^i$
$x_i \text{ lib.}$	$y^T A^i = c^i$

$$\left. \begin{aligned}
 I. \quad & (y^T A - c^T) x = 0 \\
 II. \quad & y^T (A x - b) = 0
 \end{aligned} \right\} \leftarrow$$

nel caso di coppia simmetrica !!

(sempre =)

	x_1	x_2	x_3	x_4	x_5	max
y_1	-2	1	4	0	-1	\leq 2
y_2	-1	-1	1	1	1	\leq 1
min	\leq	\leq	\leq	\leq	\leq	
	-1	3	17	5	4	

$$\Rightarrow \begin{aligned}
 \min \quad & -x_1 + 3x_2 + 17x_3 + 5x_4 + 4x_5 \\
 -2x_1 + x_2 + 4x_3 - x_5 &\leq 2 \\
 -x_1 - x_2 + x_3 + x_4 + x_5 &\leq 1 \\
 x_1, x_2, x_3, x_4, x_5 &\geq 0
 \end{aligned}$$

$$\begin{aligned}
 (-2y_1 - y_2 + 1)x_1 &= 0 \\
 (y_1 - y_2 - 3)x_2 &= 0 \\
 (4y_1 + y_2 - 17)x_3 &= 0 \\
 (y_2 - 5)x_4 &= 0 \\
 (-y_1 + y_2 - 4)x_5 &= 0
 \end{aligned}$$

$$\begin{aligned}
 (-2x_1 + x_2 + 4x_3 - x_5 - 2)y_1 &= 0 \\
 (-x_1 - x_2 + x_3 + x_4 + x_5 - 1)y_2 &= 0
 \end{aligned}$$

altrimenti posso usare uno solo

Simplesso: deve essere in forma standard!