NETWORK FLOWS

II° SEMESTRE
2021

9/03

INTRODUCTION

NETWORK FLOWS → APPROX. ALGOS → Tool: GUROBI 9.1 — Book: NETWORK FLOWS

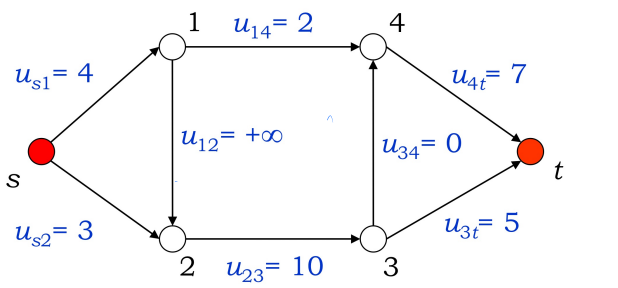
NETWORK OPTIMIZATION → NP-HARD PROBLEMS → Tool: NetworkX (Py. Library) — Book: INTEGER PROGRAMMING (L. WOLSEY, EDITION 1)

MIDTERMS
1- END MARCH
2- END APRIL
3- END MAY

NETWORK DESIGN

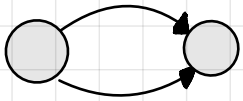
DIRECTED GRAPH · $G(N, A)$ con due nodi speciali s (SOURCE) e t (SINK)

CAPACITY · $u_{ij} \in [0, +\infty)$ è la capacità associata a $(i,j) \in A$



ASSUNZIONI

(1) G non ha un PATH DIRETTO (da s a t) di soli archi con $u_{ij} = \infty$

(2) G non ha ARCHI PARALLELI 

(3) Posso aggiungere archi con $u_{ij} = 0$ come mi pare

PATH PACKING PROBLEM

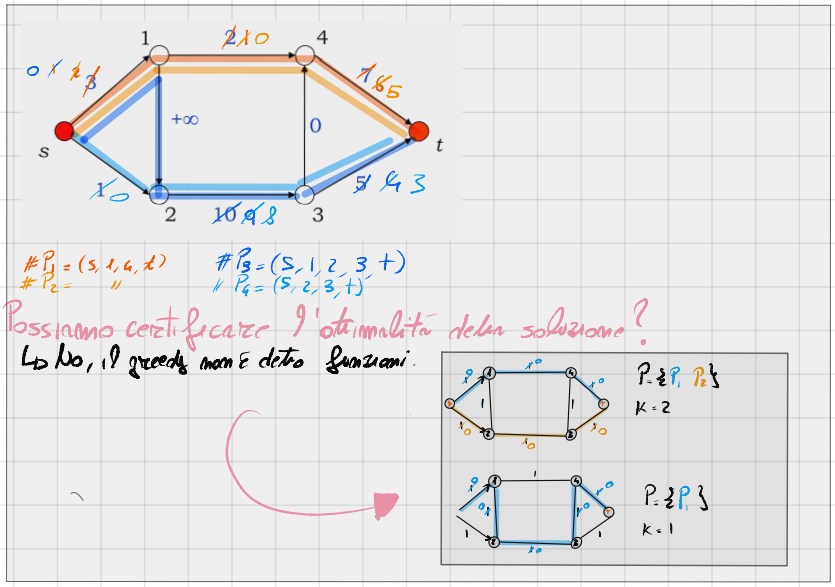
given...
· $G(N, A)$, un capacity vector $u_{ij} \in \mathbb{Z}^{|A|}$

...find
· $P = \{P_1, P_2, \dots, P_k\}$ famiglia di SIMPLE DIPATH T/C

(1) ogni $(i,j) \in A$ fa parte di AL PIÙ u_{ij} DIPATH

(2) k sia MASSIMIZZATO

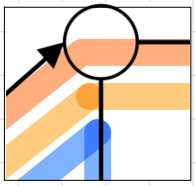
ES



ILP (FLOW FORMULATION)

VARS.

- X_{ij} occorrenze di (i,j) ∈ A nei Path in P
- ∀ v ≠ s,t => il #PATH entranti è lo stesso di quelli uscenti



✱ (TRIVIAL) OBS

NOTA: IGNORIAMO ARCHI ENTRANTI IN S

$$\max \sum_{j:(s,j)\in A} x_{sj} - \cancel{\sum_{j:(j,s)\in A} x_{js}}$$

✱ BALANCE CONSTRAINT

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 \qquad \forall i \in N \setminus \{s,t\}$$

CAPACITY CONSTRAINT

$$0 \le x_{ij} \le u_{ij}, \forall (i,j) \in A$$

INTEGRALITY REQ

$$x_{ij} \text{ Integer}, \quad \forall (i,j) \in A$$

(s,t)-FLOW
- Vettore X che soddisfa il BALANCE CONSTR.

FEASIBLE FLOW
- FLOW che soddisfa il CAPACITY CONSTR

NET FLOW IN V
- È il termine: $f_x(v) = \sum_{j:(v,j)\in A} x_{vj} - \sum_{j:(j,v)\in A} x_{jv}$

FLOW VALUE IN G
- $\delta_x(v)$

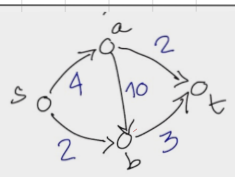
I Problemi sono equivalenti?
- Un FLOW ci dice solo le occorrenze di ogni arco, ma non ci mostra i PATH.

(trivial) =>
- Dati i Path $P = \{P_1, \dots, P_K\}$ posso facilmente costruire la soluzione al problema FF

(to prove) ⇐
- Il contrario è vero? -> Dimostrando lo proviamo l'equivalenza tra i due probl (PATH PACKING e FF)

12/03

(⇒) es



$P_1 = \{s,a,t\}$
$P_2 = \{s,a,t\}$
$P_3 = \{s,a,b,t\}$
$P_4 = \{s,b,t\}$
$k = 4$

Feasible solution to the path packing problem

⇒ Construct FF solution:

$x_{sa} = 3$ $x_{at} = 2$ $x_{ab} = 0$
$x_{sb} = 1$ $x_{bt} = 2$

- Poi controllare che soddisfa BALANCE e CAPACITY CONSTR.

Dimostriamo (⇐)

DECOMPOSITION THEOREM

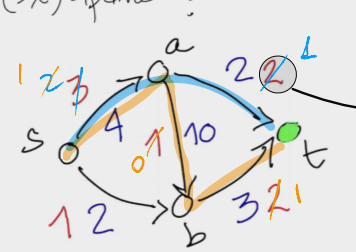
Decomposition theorem

Given a graph $G=(N,A)$, vector of capacities $u \in \mathbb{Z}_+^{|A|}$ there exists a family $\mathcal{P} = \{P_1, P_2, \dots, P_k\}$ of k (s,t) paths such that each arc $(i,j) \in A$ is used at most u_{ij} times by the paths in $\mathcal{P}$, if and only if there exists an INTEGRAL FEASIBLE (s,t) flow of value k

↳ Proof

es.

Suppose you have a feasible solution to FF. Is it always possible to build a family of k (s,t)-paths ?



(1) scegliamo un arco con $x_{it} \geq 1$

Dato che $x_{at} \geq 1$, dal Balance constraint sappiamo che c'è almeno un arco entrante

(2) prendiamo un arco (i,a) con $x_{ia} \geq 1$

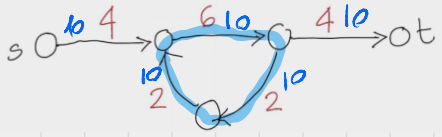
In questo caso prendiamo (s,a) e abbiamo il primo PATH

$P_1 = \{s,a,t\}$

$P_2 = \{s,a,b,t\}$

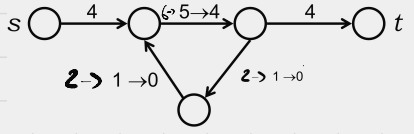
es.

Ritorna CYCLIC FEAS. FLOW



⇒

RETURN ACYCLIC FEAS. FLOW



- È possibile, dato X, FEASFlow, possiamo trasformarla in una aciclica?

CYCLIC TO ACYCLIC

- Trasformiamo X feasible flow in X' feas e acyclic flow
  - (1) Identifichiamo il ciclo
  - (2) sottraiamo 1 a ogni x_{ij} sul ciclo finché almeno un x_{ij} non diventa 0

ALGO

- (1) TRASFORMA x feas flow in X' aciclico
- (2) Partendo da t:
  - selezioniamo un arco con $x_{jt} > 0$
  - continuiamo finché s non è nel path
- (3) Ripeti (2) finché puoi

□

CUT of GRAPH

- Dato $G=(N,A)$, un TAGLIO è un insieme di archi:

$$\delta(R) = \{(v,w) \mid (v,w) \in A,\ v \in R,\ w \in \bar{R}\},\ R \subseteq N$$

NOTA

- In un grafo diretto
  - (i,j) t.c. $i \in R$ e $j \in \bar{R}$ è in $\delta(R)$
  - " $i \in \bar{R}$ e $j \in R$ è in $\delta(\bar{R})$

CAPACITY of CUT

- $u(\delta(R)) = \sum_{(i,j) \in \delta(R)} u_{ij}$

theorem WEAK DUALITY

- Per ogni (s,t)-cut $\delta(R)$ e ogni feas. flow x allora

$$x(\delta(R)) - x(\delta(\bar{R})) = \beta_x(s)$$

↳ PROOF

- Preso R, insieme che definisce un (s,t)-cut
- Considera TUTTI i BALANCE CONSTRAINT associati a $v \in R$, $v \neq s$

$$\underbrace{\sum_{j \in \delta^+(i)} x_{ij} - \sum_{j \in \delta^-(i)} x_{ji}}_{\text{LHS}} = \underbrace{0}_{\text{RHS}}$$

- Dimostreremo il teorema dividendo gli archi in 6 insiemi

ARCHI INTERNI

(1) $\forall (v,w) \mid v,w \in R$ e $v \neq s \Rightarrow x_{vw}$ NON appare in LHS del BIL. CON

(2) $\forall (v,w) \mid v,w \notin R \Rightarrow x_{vw}$ NON appare con LHS

ARCHI $R \to \bar{R}$ e $\bar{R} \to R$

(3) Per ogni $(v,w) \mid v \in R, w \notin R \Rightarrow x_{vw}$ appare in LHS con coefficiente +1

(4) Per ogni $(v,w) \mid v \notin R, w \in R \Rightarrow x_{vw}$ appare in LHS con coefficiente -1

ARCHI DA S e IN S

(5) Per ogni $(s,v) \mid v \in R \Rightarrow x_{sv}$ appare con coefficiente -1

(6) Per ogni $(v,s) \mid v \in R \Rightarrow x_{sv}$ appare con coefficiente -1

- RAGGRUPPANDO le vars che soddisfano (3),(4) otteniamo:

$$x(\delta(R)) - x(\delta(\bar{R}))$$

- RAGGRUPPANDO " " (5),(6) invece

$$- f_x(s)$$

$\Rightarrow$ In conclusione quindi:

$$\text{LHS} = x(\delta(R)) - x(\delta(\bar{R})) - f_x(s)$$

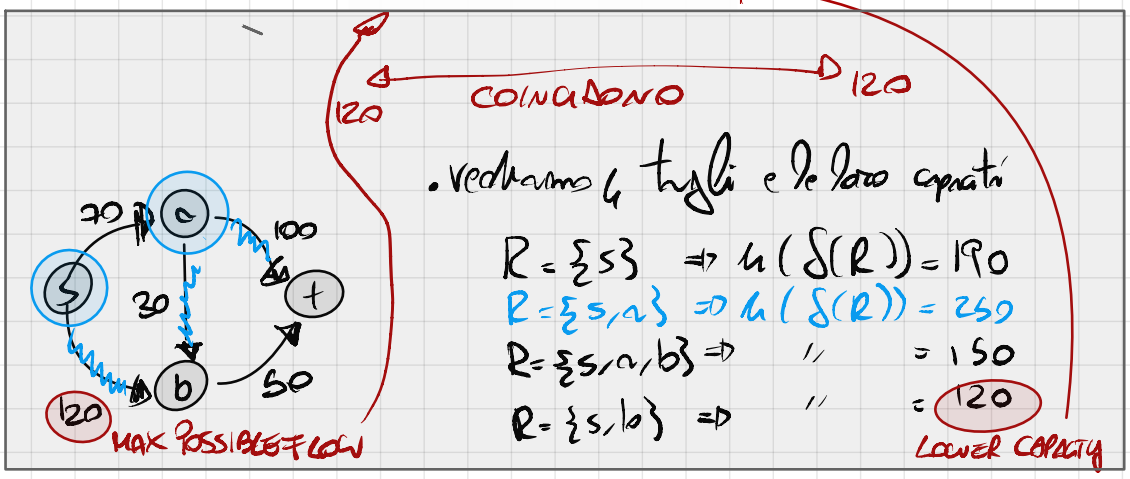
Corollary

- $\forall (s,t)$-CUT $\delta(R)$ e $\forall (s,t)$-FLOW x FEASIBLE

$$f_x(s) \leq u(\delta(R))$$

- ovviamente a interessa il CUT $\delta(R)$ con minima capacità

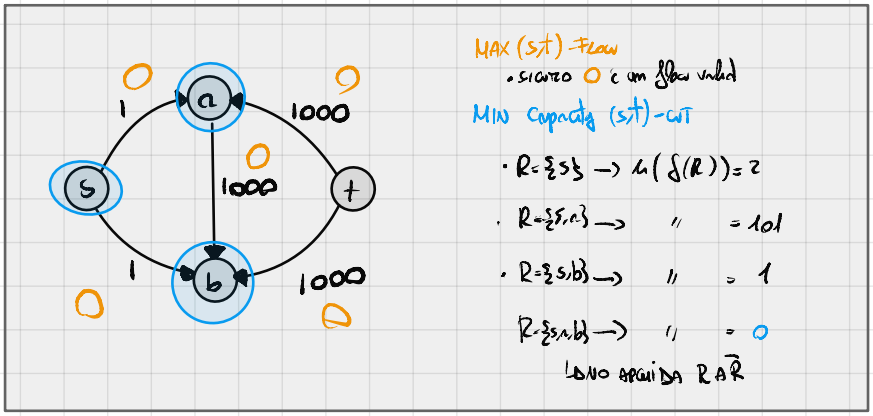
esempio



OPT. CERTIFICATE

- Con questo teorema possiamo trovare un CERTIFICATO DI OTTIMALITÀ, ovvero, grazie al WEAK DUALITY: una soluzione ottima del MIN-CAPACITY-CUT coincide con una ottima del MAX-FLOW

Esempio



Proof

- Dal WEAK DUALITY TH. sappiamo che

$$x(\delta(R)) - x(\delta(\bar{R})) = f_x(s)$$

- Dalla definizione

$$u(\delta(R)) \geq x(\delta(R))$$

$$\Rightarrow u(\delta(R)) \geq x(\delta(R)) - x(\delta(\bar{R})) \geq f_x(s)$$

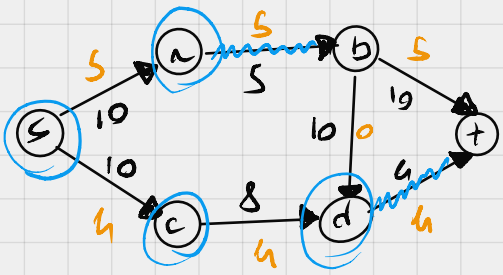
STRONG DUALITY TH.
[FORD e FULKERSON 1956 KOTIAH]

- Se $G=(N,A)$ ammette (s,t) maximum flow allora

$$\max\{f_x(s) : x \text{ is a feasible } (s,t)\text{-flow}\} = \min\{u(\delta(R)) : \delta(R) \text{ is an } (s,t)\text{-cut}\}$$

(a dire che il max flow ha sempre valore UGUALE al min capacity cut)

Esempio



- Troviamo un FLOW AMMISS.

$x = 9$

- Troviamo il min CAP. CUT

$R = \{s,a,c,b\}$

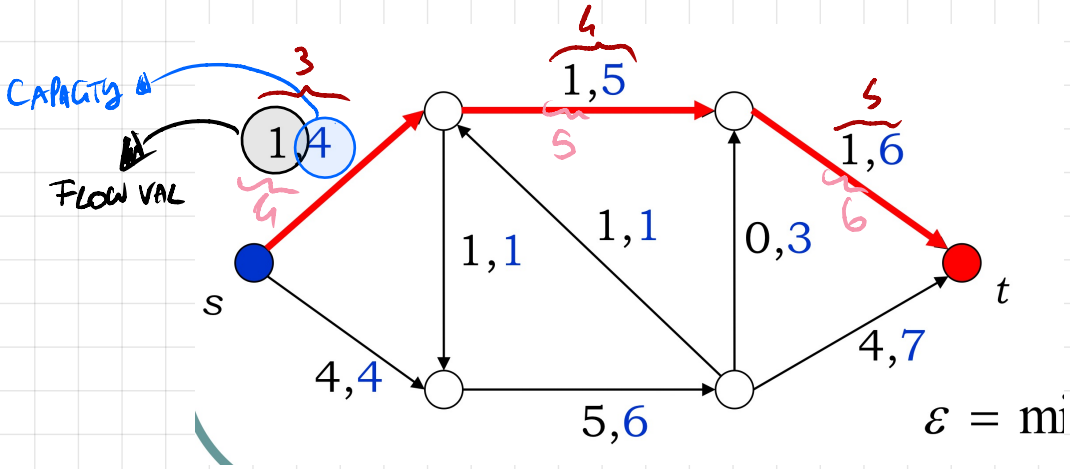
$\delta(R) = \{(a,b),(d,t)\}$

$u(\delta(R)) = 4+5 = 9$

$\Rightarrow 9=9 \Rightarrow$ il Flow è ottimo!

↳ Proof

- SE IL FLOW VAL È uguale alla CAPACITY, l'arco è SATURED
- se non è saturated possiamo incrementare il flow?



- Aggiorniamo la soluzione mettendo Flow = Flow + ε dove sta $\varepsilon = \min_{(i,j)\in P}\{u_{ij} - x_{ij}\} = 3$, quindi 1 4 che diventa 4, 4, perché il loro ε = 3
- SÌ, NON ROMPIAMO I CAPACITY CONSTRAINT, ma il BALANCE!
- Cambiamo il flow nel (s-t)-PATH di una quantità ε mantenendo, per costruzione, il Flow feasible

---

MAX FLOW (ALGO)

(0) Inizializza soluzione $x_{ij} = 0 \ \forall (i,j) \in A$

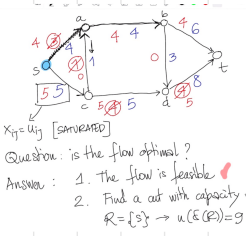
While $P \neq \emptyset$

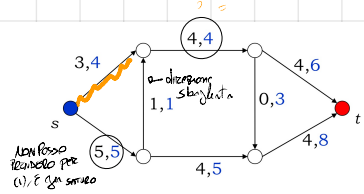
(1) cerca in G un (s-t)-PATH P t.c. $x_{ij} < u_{ij} \ \forall (i,j) \in P$

(2) incrementa il Flow x su P della quantità $\varepsilon = \min_{(i,j)\in P}\{u_{ij} - x_{ij}\}$

- TERMINATION? OPT. SOL? COMPLEXITY?

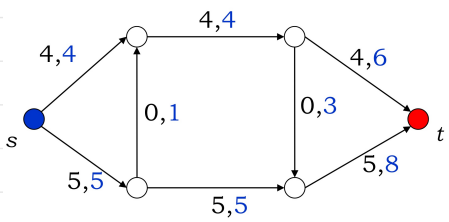
OPT. SOL.





There is no (s, t) path P such that $x_{ij} < u_{ij}$ for all arc $(i, j) \in P$, but the flow is not optimal.

① Non c'è un (s-t)-PATH che rispetti la definizione



- Come posso avvicinarmi a un optimal flow come questo nella figura accanto?

STRONG DUALITY TH. [FORD e FULKERSON 1956 KOTIAH]

↳ Proof:

FORWARD/REVERS ARCs

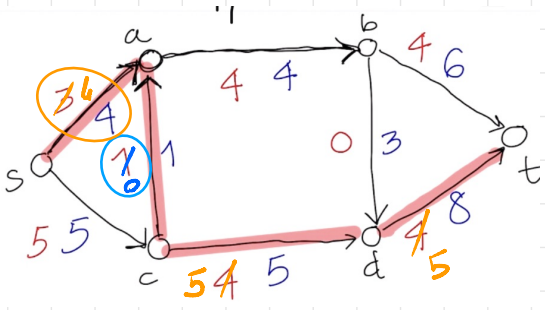
- Considera ogni $(s\text{-}t)$-PATH P, un arco può essere un
  FORWARD ARC (diretto da s a t)
  REVERSE ARC (altrimenti)

AUGMENTING PATH

- Un $(s\text{-}t)$-PATH P t/c ogni forward arc ha $x_{is} < u_{is}$ e ogni reverse arch ha $x_{ji} > 0$

$\varepsilon_1 = \min_{(i,j)\in P} \{u_{ij} - x_{ij}\} = 1$
t/c (i,j) is forward

$\varepsilon_2 = \min_{(i,j)\in P} \{x_{ji}\} = 1$
t/c (i,j) is reverse



- Se incontriamo un FORWARD SUC PATH incrementiamo il flow di ε_1, se invece incontriamo un REVERSE decrementiamo di ε_2

→ Tale tecnica ci porta da una FEAS. SOL. a un altra FEAS.

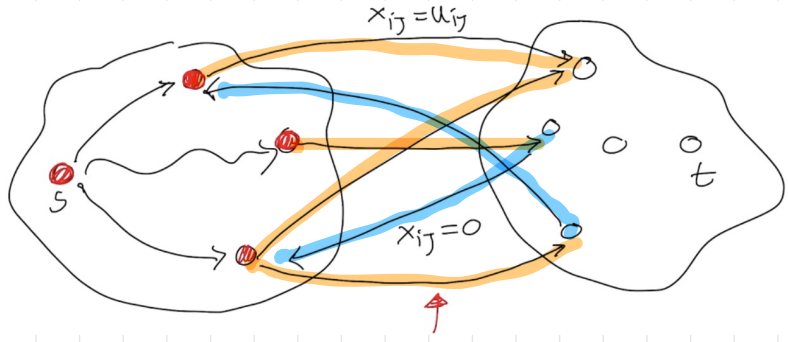
L'ALGO TROVA UNA SOLUZIONE OTTIMA?

RIC

- Weak duality TM per ottenere un certificato di ottimalità dobbiamo avere un flow e un CUT t/c $\boxed{\delta_x(s) = u(\delta(R))}$

- SUPPONIAMO MAX FLOW x

> I NODI ROSSI SONO QUELLI RAGGIUNGIBILI TRAMITE AUGMENTING PATH
> I BIANCHI NO
> GLI ARCHI IN MEZZO SONO FORWARD O REVERSE
>
> $\begin{cases} x_{is} = u_{ij}, \text{ per i forward} \\ x_{ji} = 0, \text{ per i reverse} \end{cases}$



$$\delta(s) = \underbrace{x(\delta(R))}_{u(\delta(R))} - \underbrace{x(\delta(\bar{R}))}_{=\emptyset} = u(\delta(R))$$

FORD e FULKERSON ALGO

(1) INIZIALIZZA il flow $X=0$

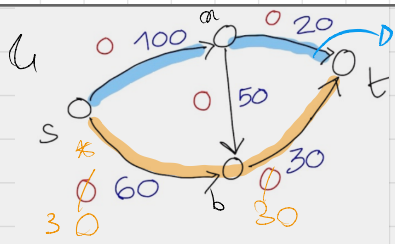
DO

(2) Cerca (s,t)-AUGMENTING PATH P

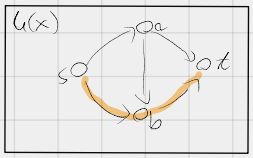
(3) Incrementa il flow sul path di $X \min\{\varepsilon_1, \varepsilon_2\}$

WHILE $P \neq 0$

esempio



costruiamo un AUXILIARY GRAPH $G(x)$

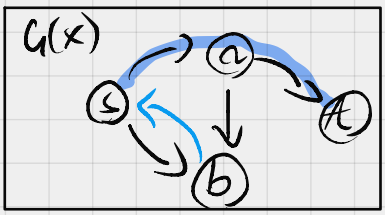


- Cerchiamo un oriented (s,t)-path
- Usiamo questo sul grafo di partenza e incrementiamo il flow *

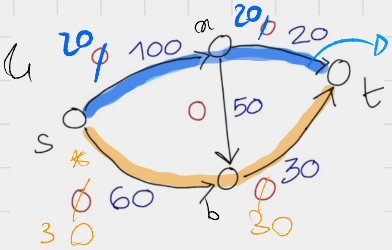
- Dopo aver trovato l'ORIENTED PATH E INCREMENTATO IL FLOW CALCOLIAMO $\delta_x(s) = u(\delta(R))$

→ Disegnamo il grafo con solo gli archi NON-SATURATI

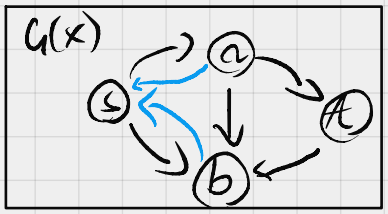
→ Possiamo aggiungere archi inversi dove il flow è $x_{ij} > 0$



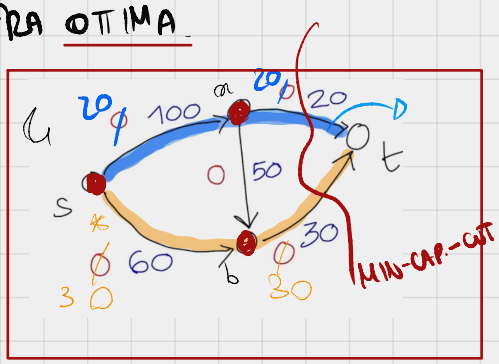
- TROVIAMO UN ORIENTED (s,t)-PATH
- AGGIUNGIAMOLO AL GRAFO SCALANDO i FLOW



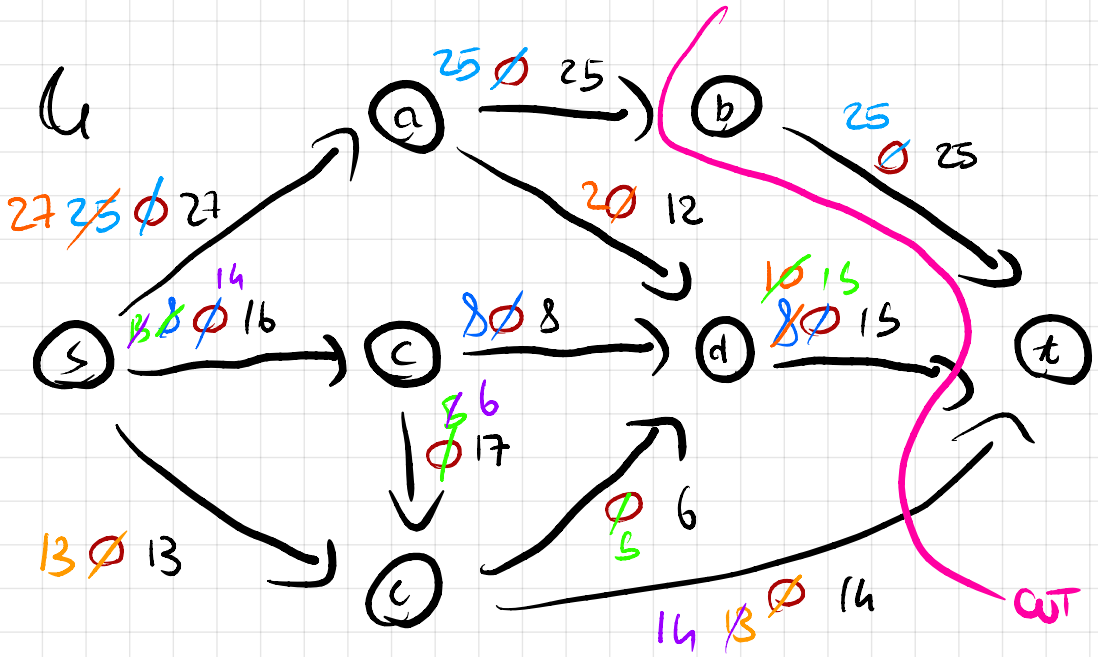
- Continuiamo
- non c'è più un AUGMENTING PATH.



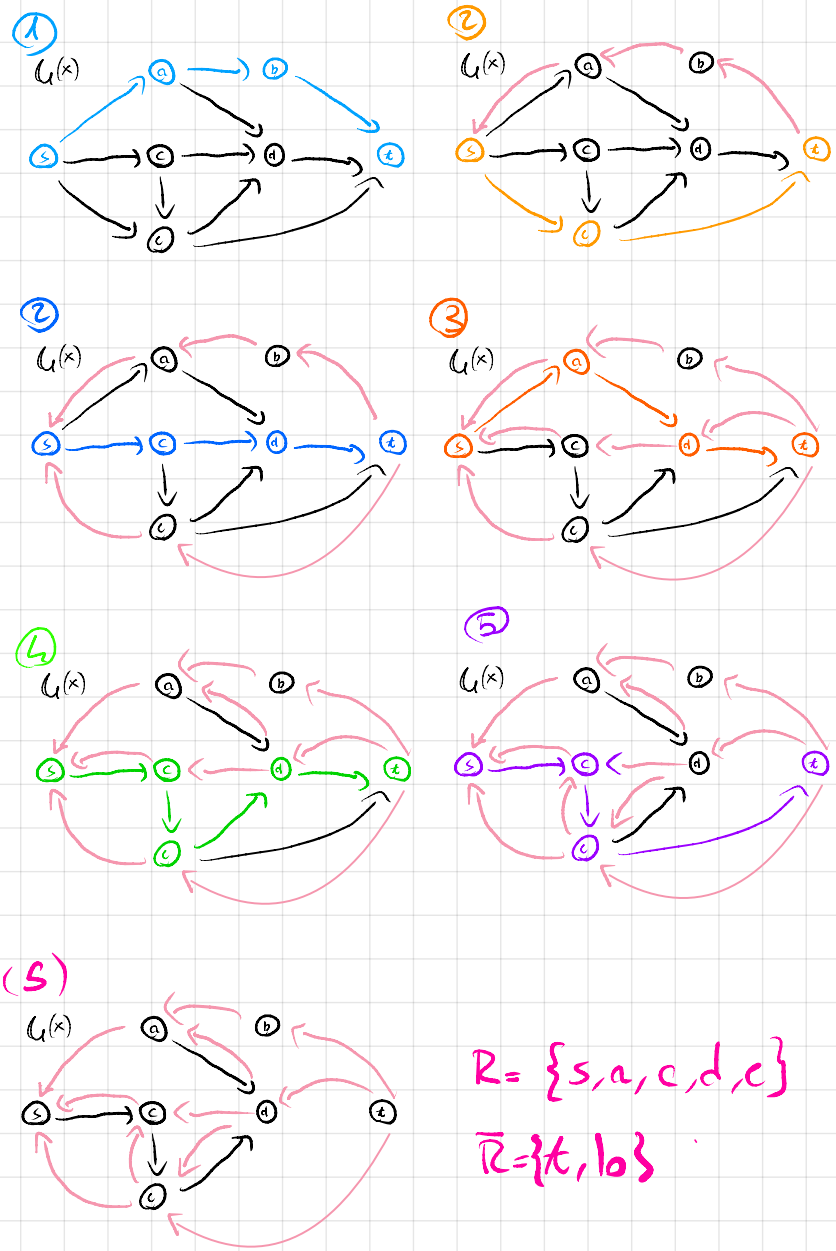
⇒ L'ULTIMA SOLUZIONE ERA OTTIMA



esempio



$\Rightarrow \delta_x(s) = 25 + 38 + 8 + 2 + 5 + 1 = 79$



$R = \{s, a, c, d, e\}$

$\overline{R} = \{t, b\}$

ooo SOMMARIO

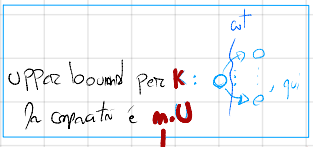
Max flow - min Cut

- se un grafo $G=(N,A)$ ammette un (s,t) maximum flow allora $\max\{\beta_x(s) \mid x \text{ è una FEAS. SOL.}\} = \min\{u(\delta(R)) \mid \delta(R) \text{ è un } (s,t)\text{-CUT}\}$

Ford and Fulkerson (ALGO)

(è feasible per il Th sopra) (termina per quello sotto)

- Inizializzazione (Un qualsiasi flow ammissibile)
  $x=0$
  DO
    Cerca in G un (s,t)-AUGMENTING PATH
    Incrementa il flow su P di $x \min\{\varepsilon_1, \varepsilon_2\}$
  WHILE $P \neq \emptyset$



$K \Rightarrow O(Knm) \leq O(mU_K)$

TH

- Un feasible flow x è ottimo $\Leftrightarrow$ non ci sono augmenting path in G

EDMONDS AND KARP

- Se a ogni step di F&F scegliamo lo SHORTEST AUG. PATH, la complessità è $O(nm^2)$ Perché lo visitiamo con un BFS

- COME RAGGIUNGIAMO UNA COMPLESSITÀ POLINOMIALE?
  ↳ L'unico modo di farlo è sfruttare le giuste istanze del problema:
  (1) Perché partire da $x=0$?
  (2) seleziona l'augmenting path che porta al massimo incremento del flow

---

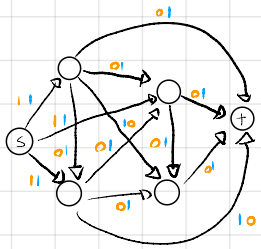
23/03

ESERCIZI PARZIALE

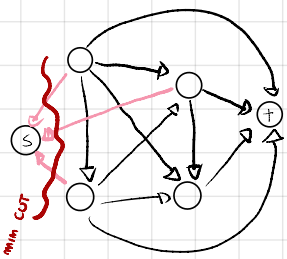
(1)

- Sia $G=(N,A)$ un grafo diretto
- Trova il minimo numero di archi la cui rimozione disconnetta s da t

➡
- Risolviamo trovando il max flow con F&F in modo da avere anche il min CUT
- Possiamo iniziare settando le CAPACITY a 1



TROVIAMO UN FLOW FEASIBLE:
↳ $\beta_x(s)=3$
↳ x è feasible



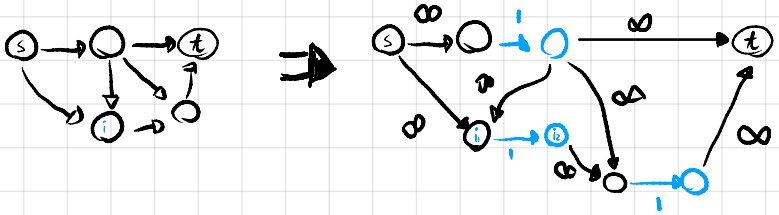
- Alla prima iterazione non troviamo alcun AUGM. PATH, quindi x è massimo e corrisponde al min CUT

---

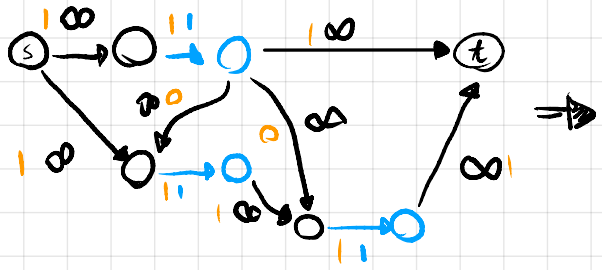
(2)

- Sia $G=(N,A)$ un grafo diretto e s,t due nodi di N
- Trova il minimo numero di nodi la cui rimozione disconnetta s da t

➡
- Node splitting raddoppio tutti i nodi tranne s,t e inizializzo le capacity a ∞ sui vecchi archi e 1 sui nuovi

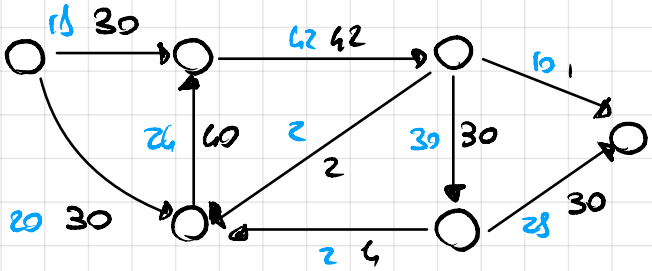


- Vediamo col flow



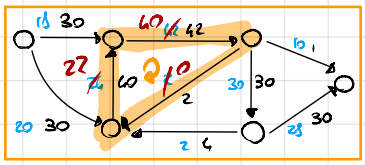
---

(3)



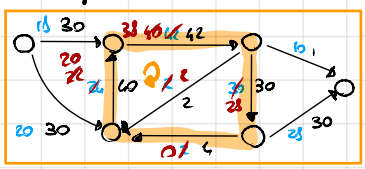
(1) · Il flow è feasible?
(2) · Ci sono flow equivalenti che non inducono un ciclo?
(3) · Trova il massimo numero di (s,t)-PATH che possono essere PACKED

1↳ Capacity e Balance constrs soddisfatti
2↳ Ecco il ciclo

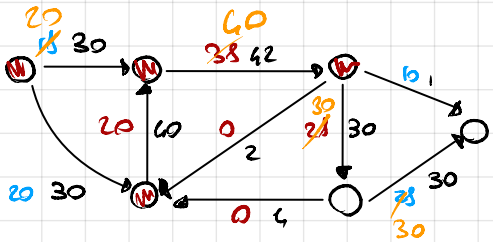


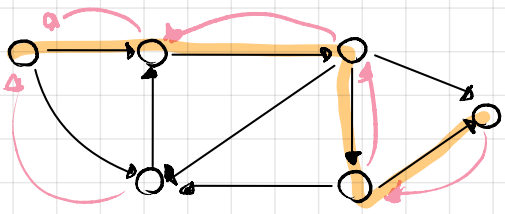
Applichiamo la tecnica discussa nella prova del TH.1
→ RIDUCIAMO IL FLOW sul ciclo fino a portare a $\emptyset$ il flow dell'arco che lo aveva più piccolo

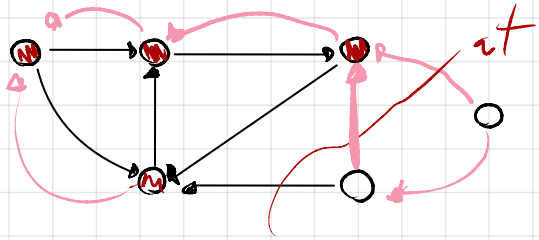
Lo spostiamo in un altro ciclo, ancora RIPETIAMO.



3↳ Applichiamo max (s,t)-Flow

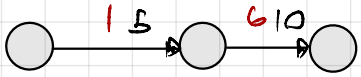






FLOWS WITH LOWER BOUNDS

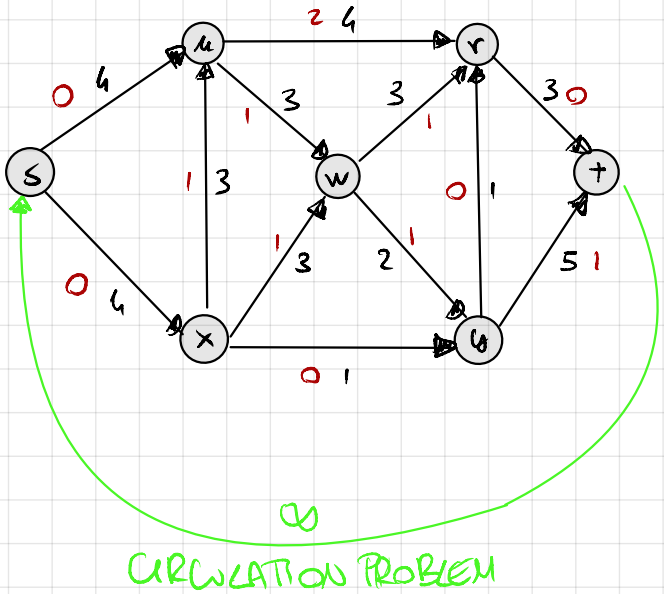
- Se impostiamo dei requisiti di minimo sul flow, esiste sempre una FEAS. SOL. ?



// NON ESISTE PERCHÉ IL MINIMO FLOW RICHIESTO DA (b,c) È 6, E IL MASSIMO SU (a,b) È 5

FEASIBILITY CHECK

#MIN-REQs: 9_{ts}



∞

CIRCULATION PROBLEM

- BALANCE CONSTRAINTS

$$
\begin{aligned}
s:\ & x_{su} + x_{sx} - x_{ts} = 0\\
u:\ & x_{uv} + x_{uw} - x_{su} - x_{xu} = 0\\
v:\ & x_{vt} - x_{uv} - x_{wv} - x_{yv} = 0\\
w:\ & x_{wv} + x_{wy} - x_{uw} - x_{xw} = 0\\
x:\ & x_{xu} + x_{xw} + x_{xy} - x_{sx} = 0\\
y:\ & x_{yv} + x_{yt} - x_{wy} - x_{xy} = 0\\
t:\ & x_{ts} - x_{vt} - x_{yt} = 0
\end{aligned}
$$

- BOUNDS DI x_{ij} → $\boxed{\ell_{ij} \le x_{ij} \le u_{ij}}$
- Sostituiamo tutti gli x_{ij} → $\boxed{x_{ij} = x'_{ij} + \ell_{ij}}$

$$\Rightarrow \ell_{ij} \le x_{ij} \le u_{ij} \Rightarrow \ell_{ij} \le x'_{ij} + \ell_{ij} \le u_{ij} \Rightarrow \boxed{\overbrace{\ell_{ij} - \ell_{ij}}^{0} \le x'_{ij} \le u_{ij} - \ell_{ij}}$$

$x_{ij} = x'_{ij} + \ell_{ij}$

- i BALANCE CONSTRs diventano

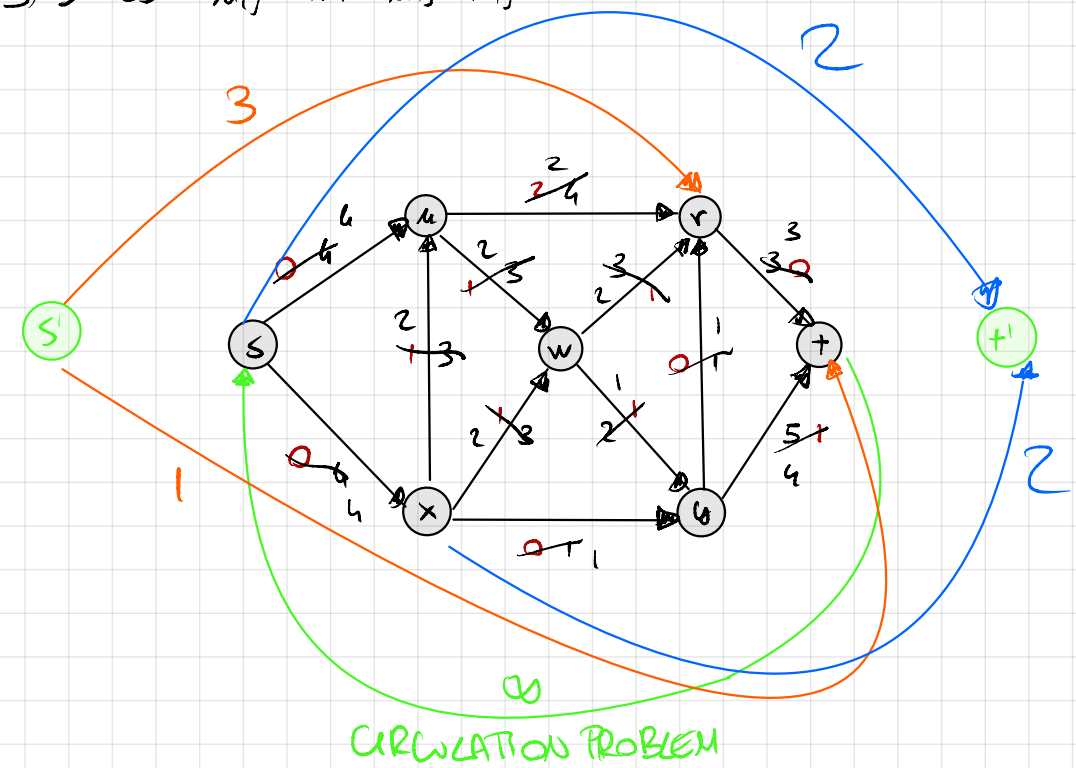
$$
\begin{aligned}
s:\ & x_{su} + x_{sx} - x_{ts} = 0\\
u:\ & x_{uv} + x_{uw} - x_{su} - x_{xu} = 0 - 2 \rightarrow u:\ x'_{uv} + \ell_{uv} + x'_{uw} + \ell_{uw} - x'_{su} - \ell_{su} - x'_{xu} - \ell_{xu} = 0\\
v:\ & x_{vt} - x_{uv} - x_{wv} - x_{yv} = 0 - 3 \Rightarrow x'_{uv} + 2 + x'_{uw} + 1 - x'_{su} - 0 - x'_{xu} - 1 = 0\\
w:\ & x_{wv} + x_{wy} - x_{uw} - x_{xw} = 0 \Rightarrow x'_{uv} + x'_{uw} + x'_{su} = -2\\
x:\ & x_{xu} + x_{xw} + x_{xy} - x_{sx} = 0\\
y:\ & x_{yv} + x_{yt} - x_{wy} - x_{xy} = 0\\
t:\ & x_{ts} - x_{vt} - x_{yt} = 0
\end{aligned}
$$

- Ora trasformiamo ciò che abbiamo in un FLOW PROBLEM

(1) Per ogni nodo i con $RHS_i > 0$ aggiungi un arco (s', i) con $u_{s'i} = RHS_i$

(2) Per ogni nodo j con $RHS_j < 0$ aggiungi un arco (j, t') con $u_{jt'} = -RHS_j$

(3) SCALO u_{ij} con $u_{ij} - x_{ij}$

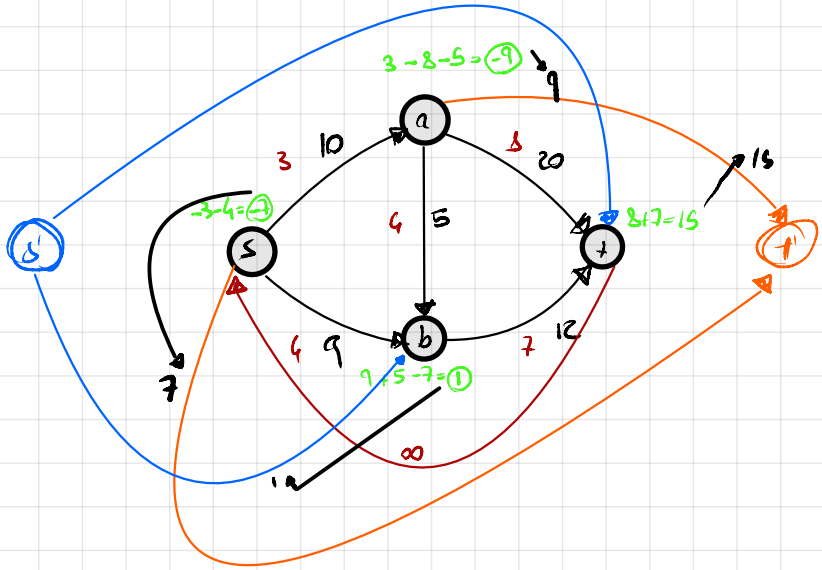


$\Rightarrow$ s' e t' sono connessi a UNBALANCED NODE, cioè con $RHS \neq 0$

- Se troviamo ora a trovare un (s', t')-PATH che satisfi gli archi nuovi, troviamo un feasible flow per il problema originale
- Se non troviamo $\Rightarrow \nexists$ FEAS SOL.

#BALANCES

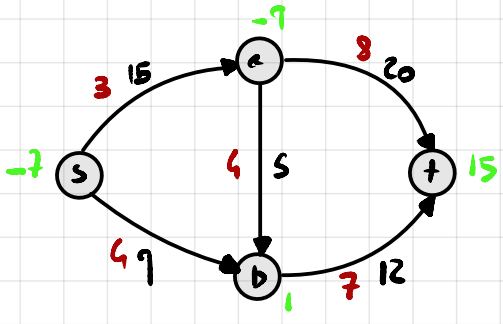
$\#q_{15}$



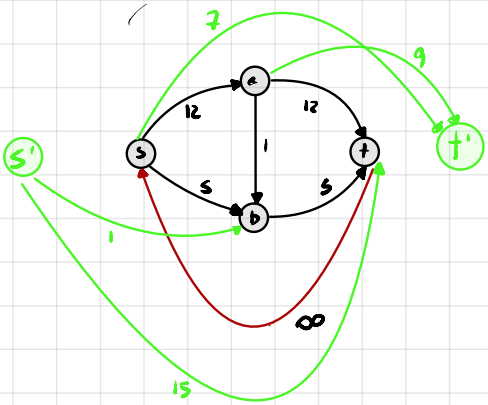
- ORA BASTA APPLICARE F&F, se non troviamo $\Rightarrow$ G non ammette feas. flow

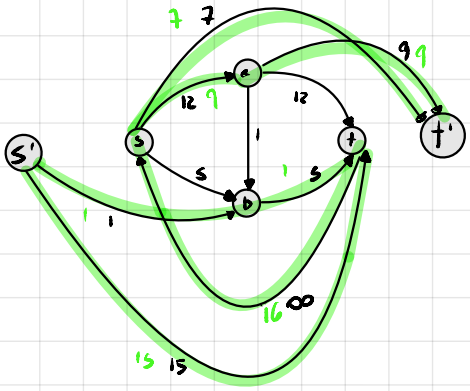
ESERCIZIO

- ΣENTRANTI - ΣUSCENTI



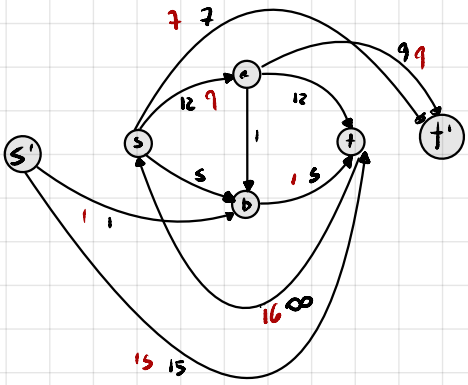
- INTRODUCIAMO s', t';
- RISCALIAMO LE CAPACITÀ COME DIFFERENZA TRA CAPACITÀ e LBs



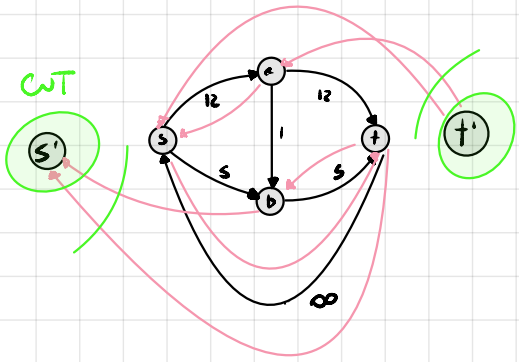


- INIZIALIZZIAMO IL FLOW

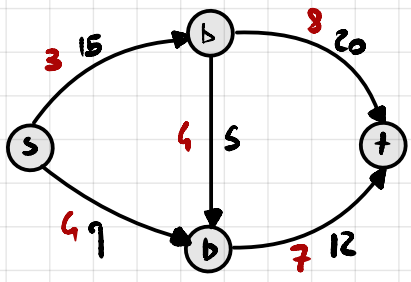
- APPLICHIAMO F&F



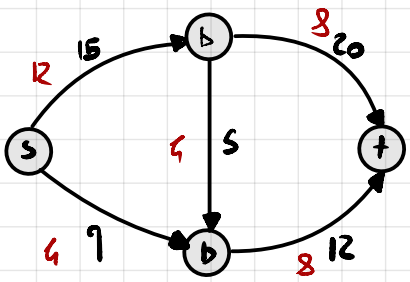
⇒



- $f_x(S) = u(\delta(R))$ ✓ ⇒ IL PROBLEMA AMMETTE FEAS. FLOW
- Torniamo al grafo iniziale e vediamo se ha una feas. solution:

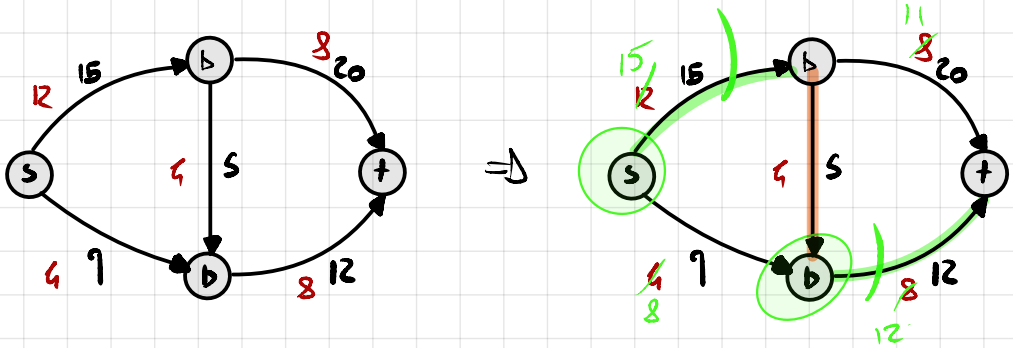


⇒



✓

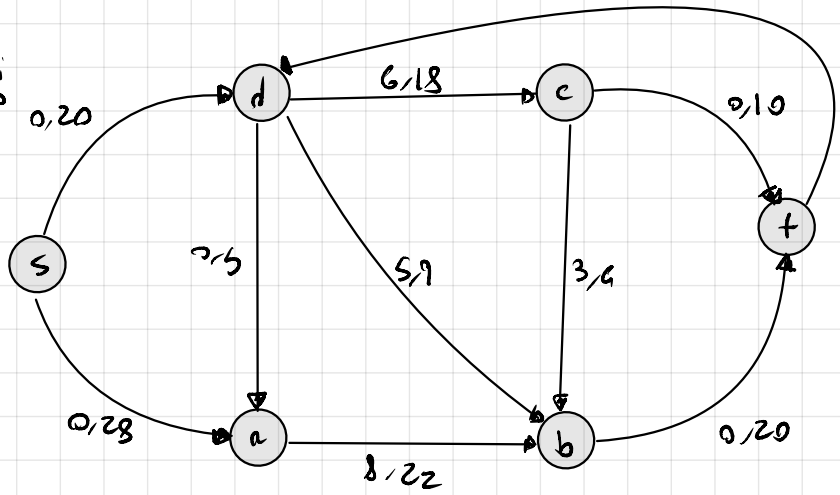
• Ora con FF troviamo il sol ottima



$\Rightarrow f_x(S) = 15 + 8 = u(\delta(R)) - l(\delta(\bar{R})) = 15+12-4$

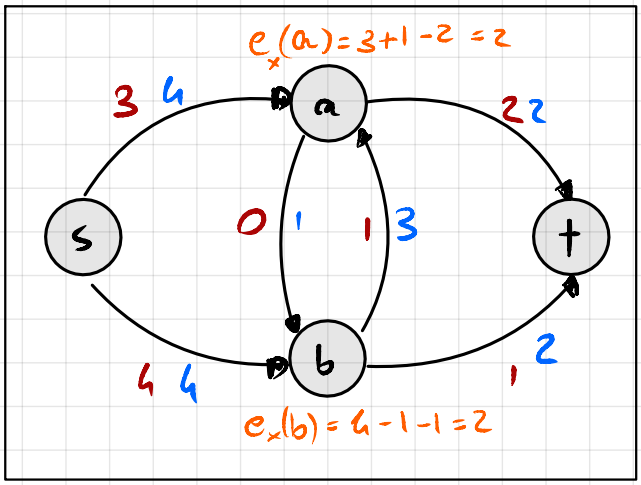
COM. BOUNDS

Dato:



(1) Valuta il max (s-t) - flow

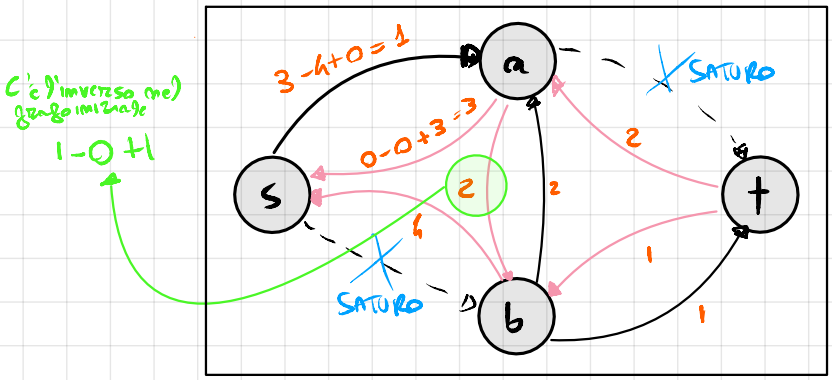
(2) Possiamo incrementare il max (s-t) - flow by adding one arc tra una coppia di nodi che non sia (s,t)



- Non è un FEASIBLE FLOW
- Se una FLOW DISTRIBUTION x ha:
  (1) $e_x(u) \geq 0 \quad \forall u \in N, u \neq \{s,t\}$ — VIOLA IN ECCESSO IL BILANCIO
  (2) $0 \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in A$ — SODDISFA LA CAPACITÀ

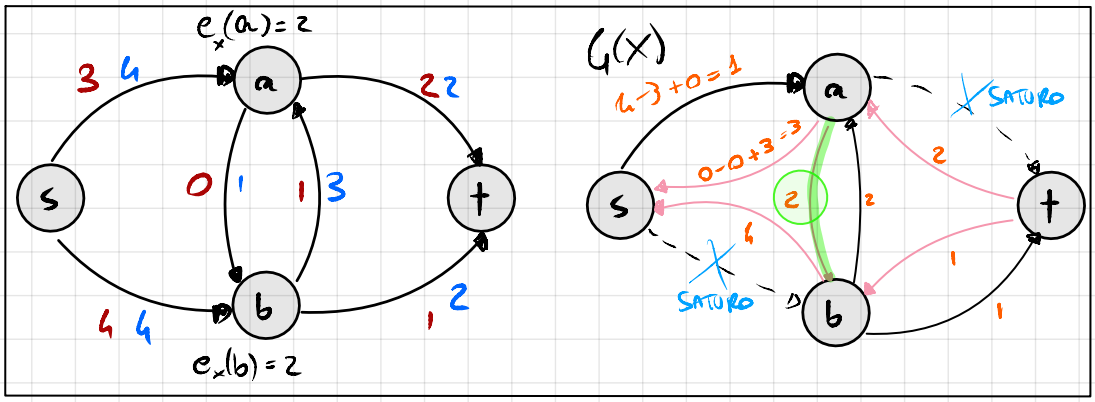
⇒ Lo chiamiamo FEASIBLE PREFLOW

- Possiamo associare un RESIDUAL GRAPH $G(x)$ ha un preflow
  - $G(x)$ ha un arco $(i,j) \Leftrightarrow x_{ji} > 0$ OR $x_{ij} < u_{ij}$

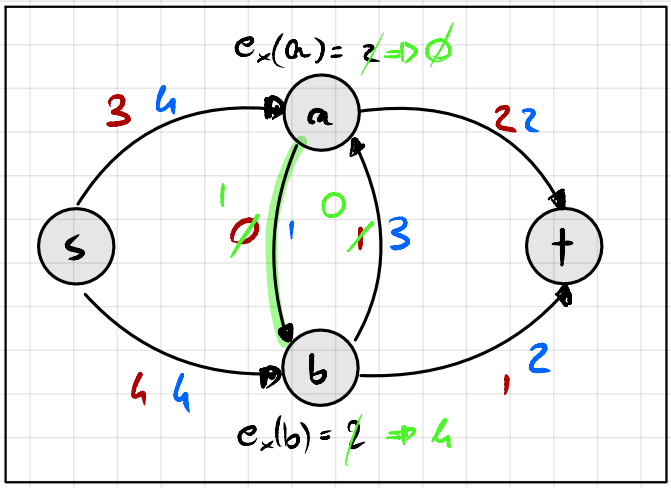


  - La nuova capacity $\bar{u}_{ij} = u_{ij} - x_{ij} + x_{ji}$, cioè il flow che può mancare per saturare l'arco
    - È valido anche per gli archi inversi, cioè il flow dell'arco inverso nel flow iniziale

---



- L'arco (b, a) in $G(x)$ ci permette di pushare due unità di flow, quindi:

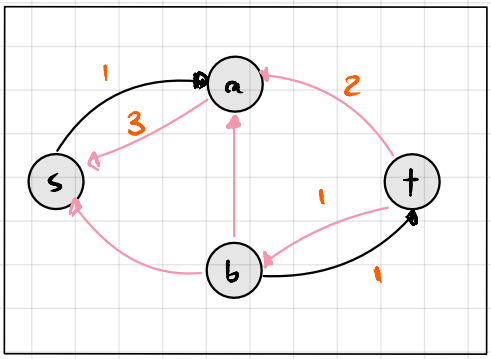


- L'operazione di PUSH modifica gli eccessi di flow nelle coppie di nodi coinvolti

↳ FEAS. PREFLOW → FEAS. PREFLOW!

- Se tutti gli $c_x(u) = 0$ → FEAS. FLOW!

- Possiamo continuare a creare $G(x)$ e pushare finché tutti gli $c_x(u)$ non sono uguali a 0



ACTIVE NODE

- Un nodo $i \in N$ è ATTIVO se $e_x(i) > 0$

VALID LABELLING

- Un vettore $d \in \mathbb{Z}_+^{|N|}$ è detto VALID LABELLING per un PREFLOW se:

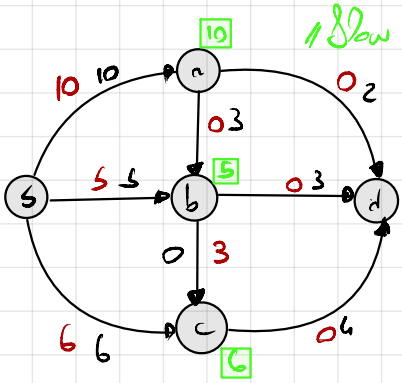
(1) $d(s) = n = |N|$

(2) $d(t) = 0$

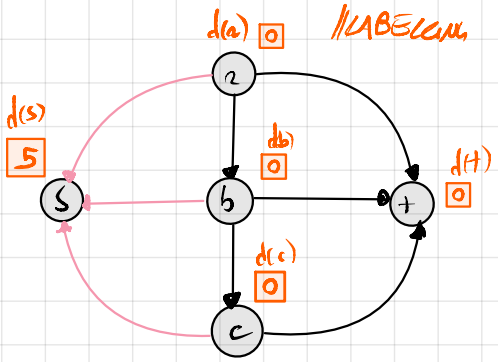
(3) $\forall (i,j) \in \bar{A} \mid \bar{A}$ archi di $G_f(x) \Rightarrow d(i) \leq d(j) + 1$

- Dato $G=(N,A)$ è sempre possibile identificare un PREFLOW e un VALID LABELLING (ma non tutti i preflow ammettono un VALID LABELLING)

Per trovare un preflow saturiamo tutti gli archi uscenti da s



flow postabile



- Check (3) → $d(i) \leq d(j)+1$ ⇒
$$\begin{cases} d(a) \leq d(s)+1 \\ d(a) \leq d(b)+1 \\ d(a) \leq d(c)+1 \\ \vdots \end{cases}$$

THEOREM

- se X è 1) FEAS. PREFLOW e d è un VALID LABELLING per X
⇒ $\exists$ (s,t)-cut R t.c. $x_{is} = u_{is}$ $\forall x_{is} \in \delta(R)$ e $x_{is}=0$ $\forall (i,j) \in \delta(\bar{R})$

L11_PT2

PUSH LABEL ALGORITHM

INIZIALIZZA: Preflow X, Labels d, Eccessi e

WHILE (X NOT FEAS FLOW)

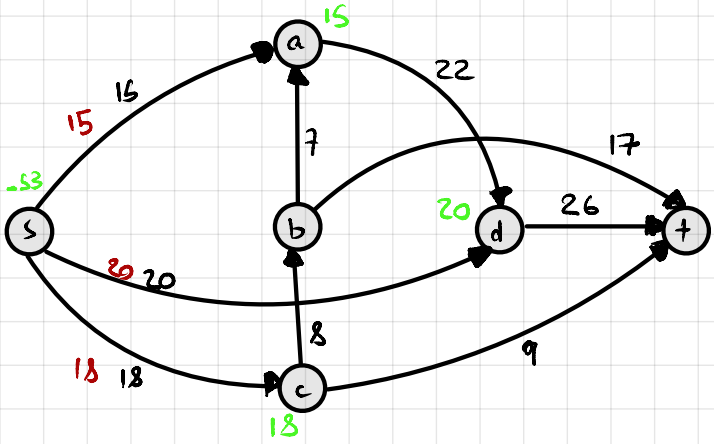
  Scegli NODO ATTIVO i (con $e>0$) su $G(X)$

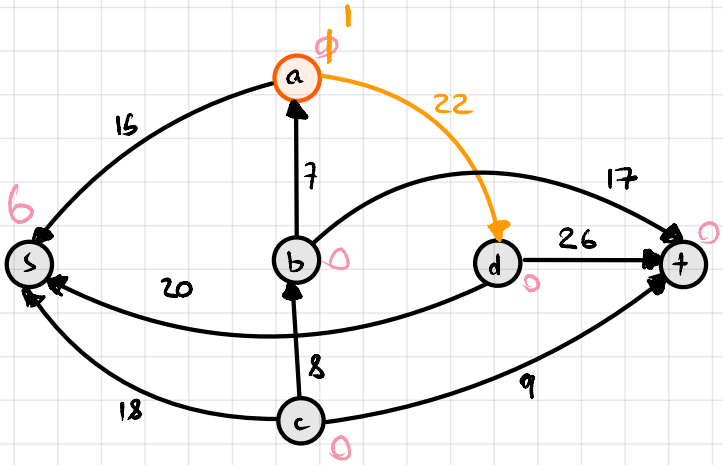
  WHILE ($\exists$ ARCO AMMISSIBILE (i,j) t.c. $d_i = d_j + 1$)

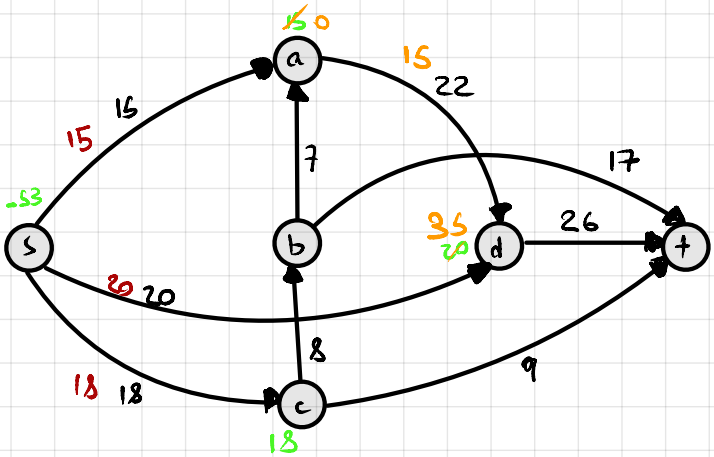
    PUSH (i,j)

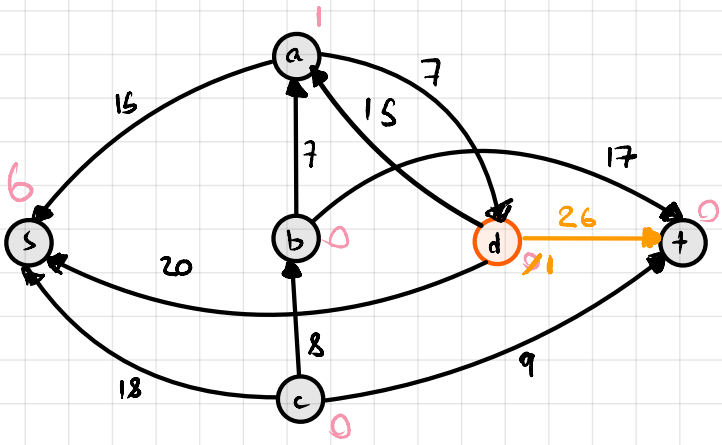
  IF (i è ancora attivo)

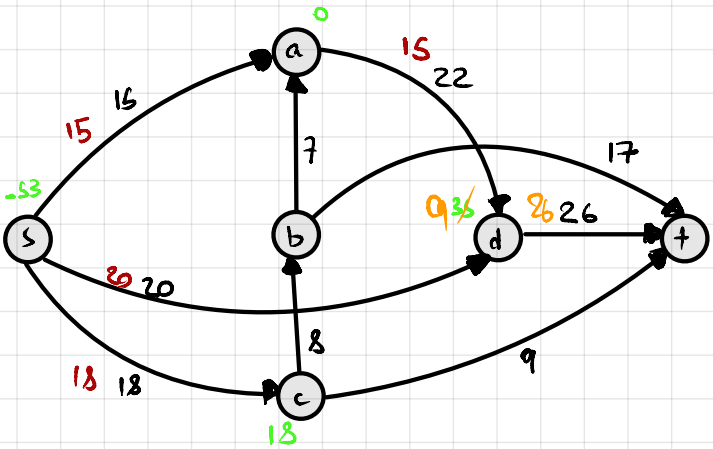
    RELABEL(i)

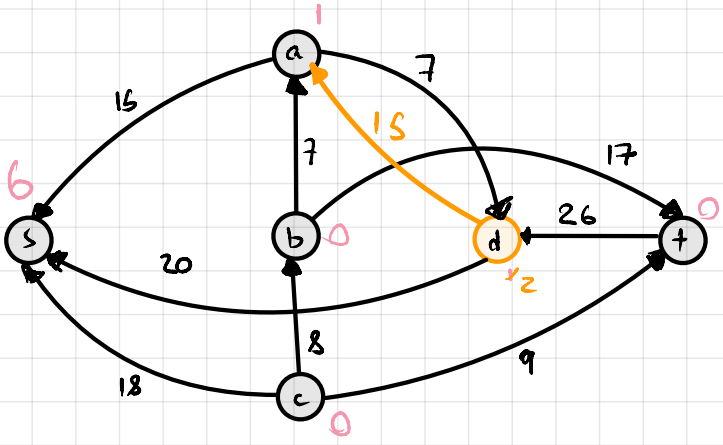


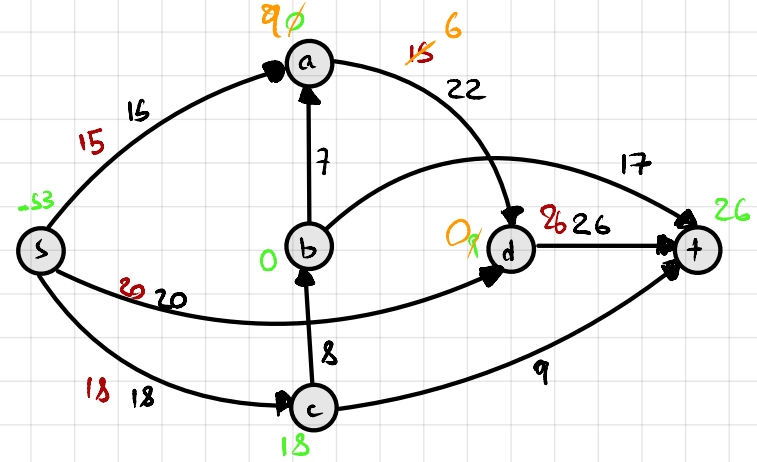


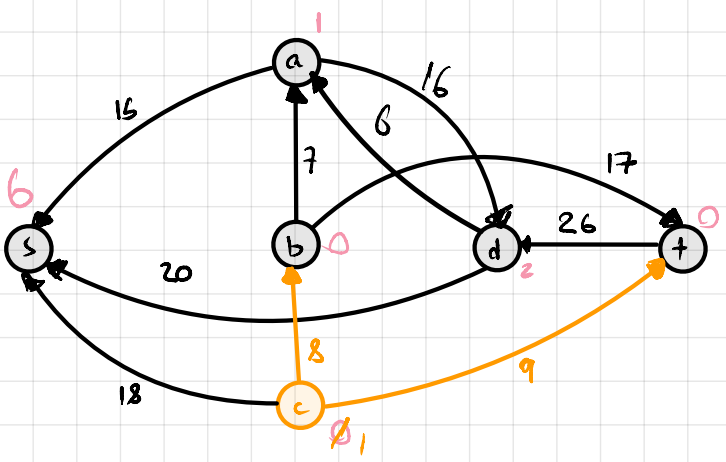


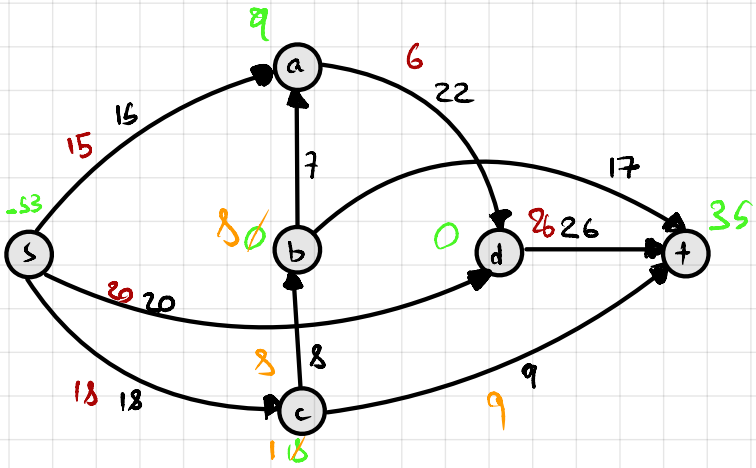


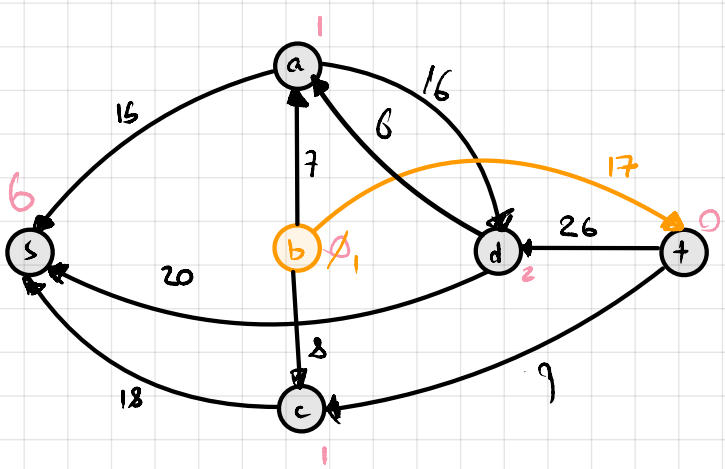


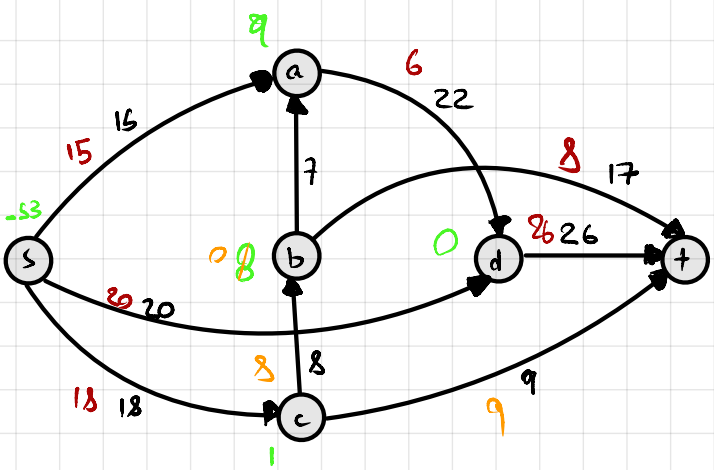


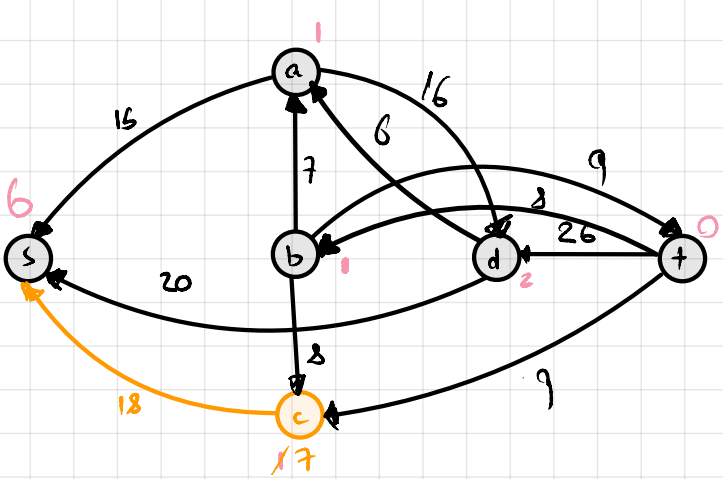

s
a
b
c
d
t
15
15
22
7
17
26
20
18
18
8
9
-53
26
18

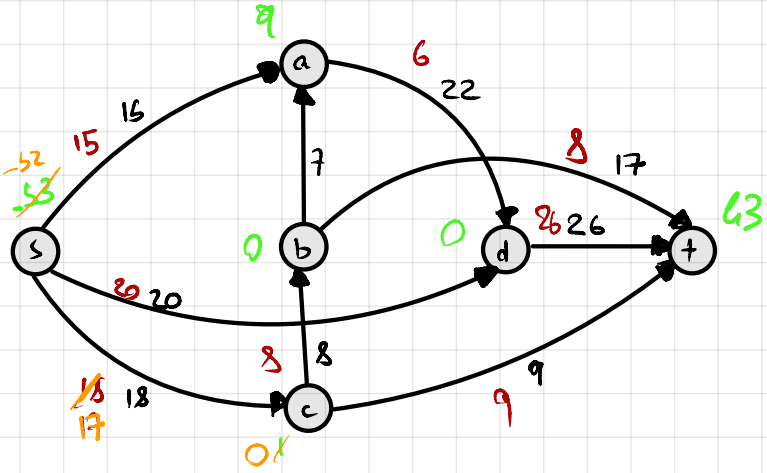

s
a
b
c
d
t
15
16
6
7
17
26
20
18
8
9
6
26
0

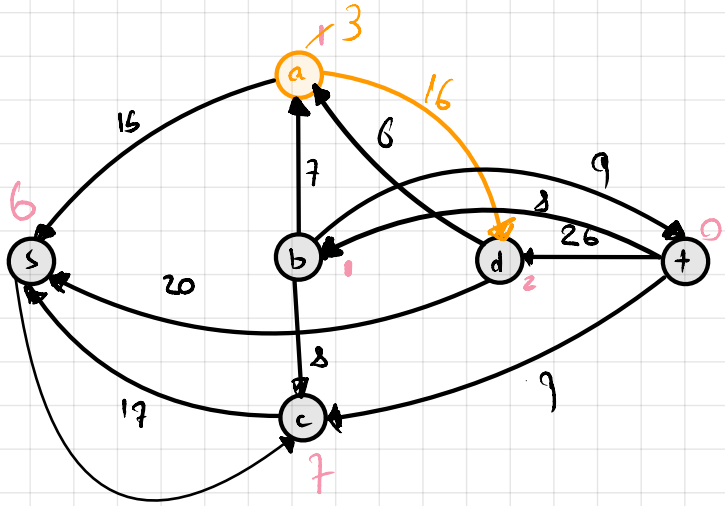

s
a
b
c
d
t
15
15
22
7
17
26
20
18
18
8
9
-53
26
35


s
a
b
c
d
t
15
16
6
7
17
26
20
18
8
9
6
26
0


s
a
b
c
d
t
15
15
22
7
17
26
20
18
18
8
9
-53
26
43

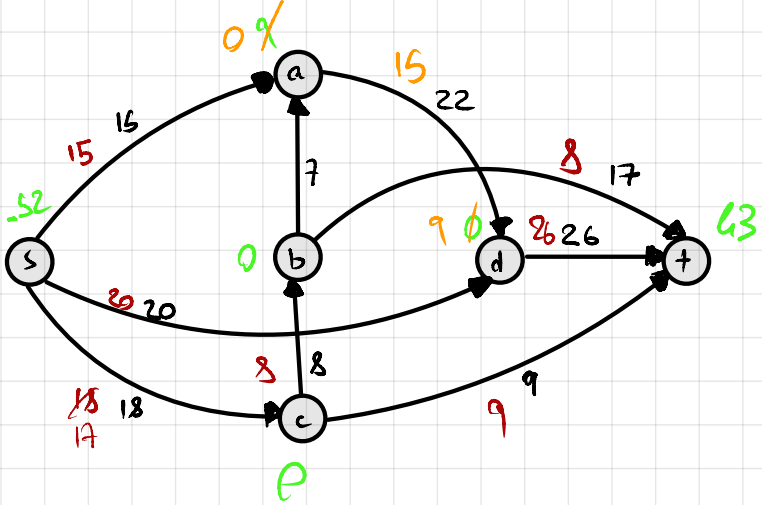

s
a
b
c
d
t
15
16
6
7
17
26
20
18
8
9
6
26
0

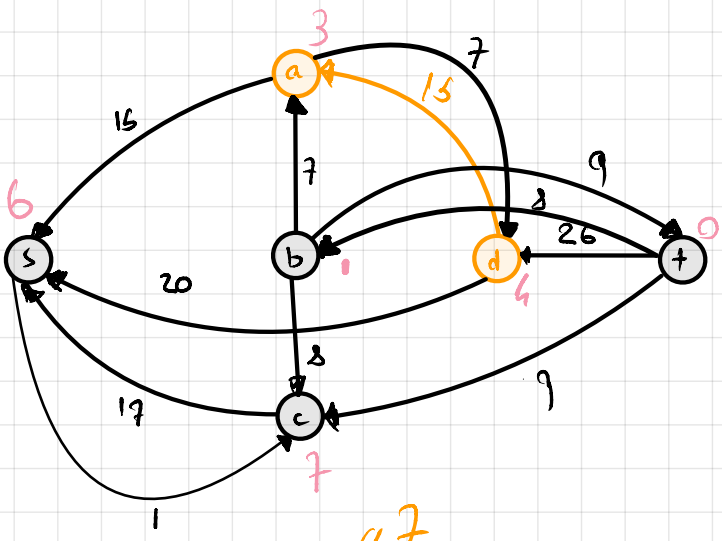

s
a
b
c
d
t
15
15
22
7
17
26
20
18
18
8
9
-53
26
8

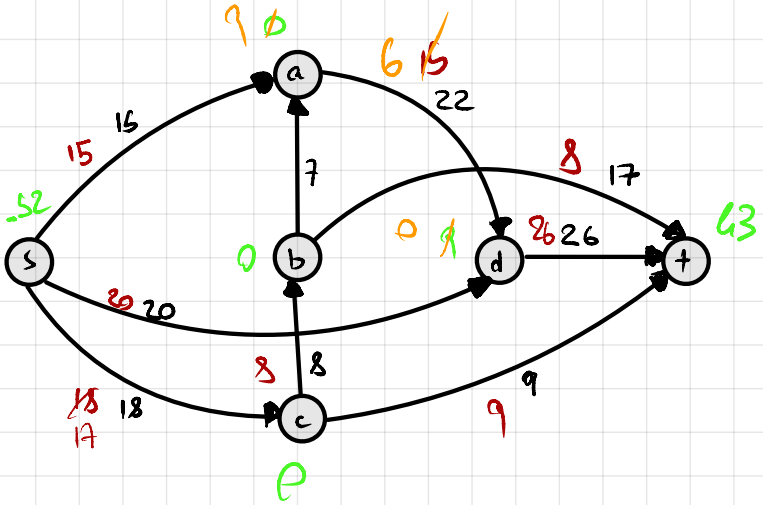

s
a
b
c
d
t
15
16
6
7
17
26
20
18
8
9
6
26
0

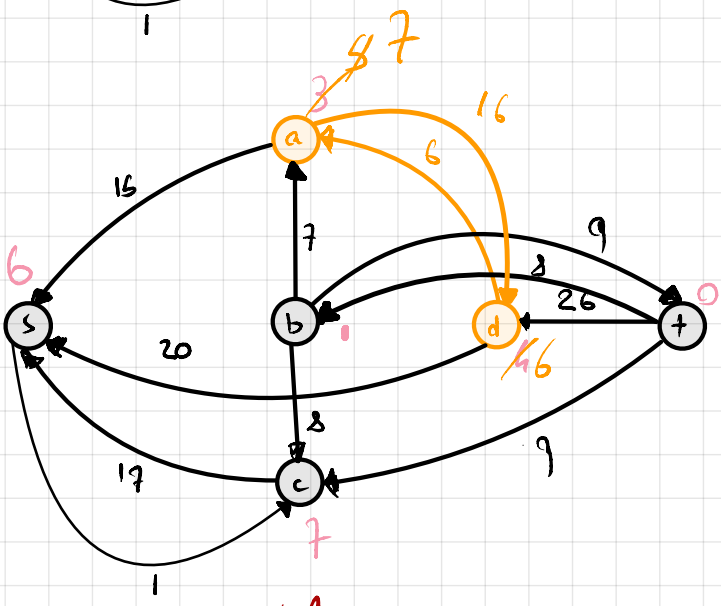
s
a
b
c
d
t
15
15
22
7
17
26
20
18
18
8
9
-52
-53
26
43

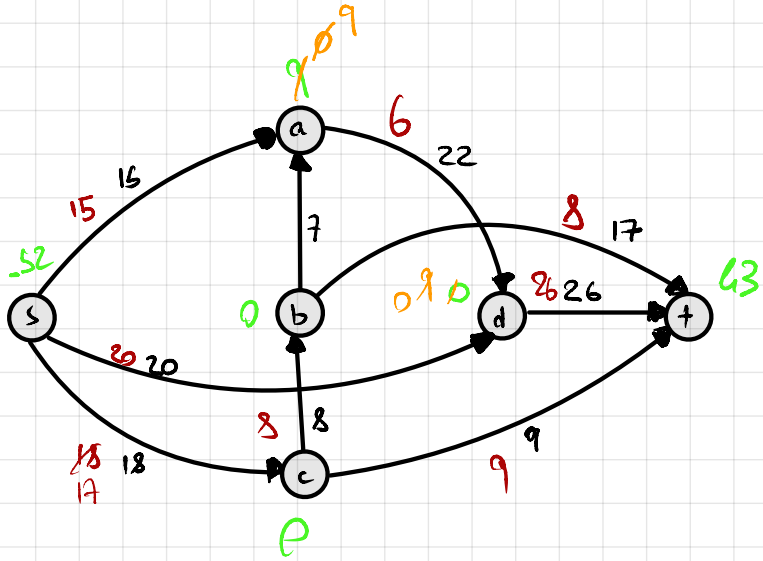
s
a
b
c
d
t
15
16
6
7
17
26
20
18
8
9
6
26
0

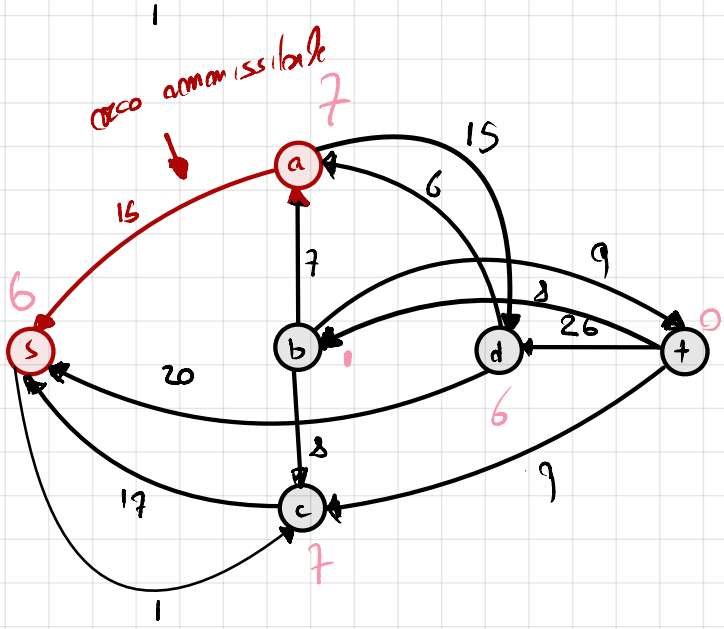


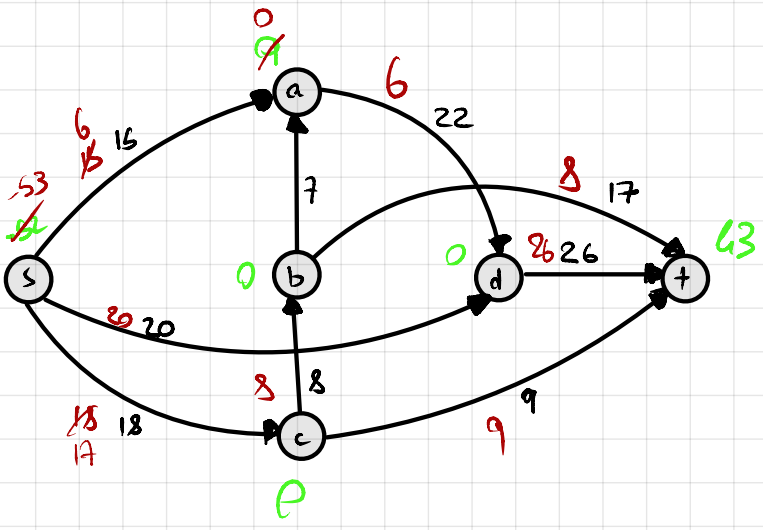












⇒FEASIBLE FLOW

MINIMUM CUT

Distribuisci 3 moduli su 2 processori, se due moduli sono su processori differenti, paghi un COMMUNICATION COST

ASSIGNMENT COST

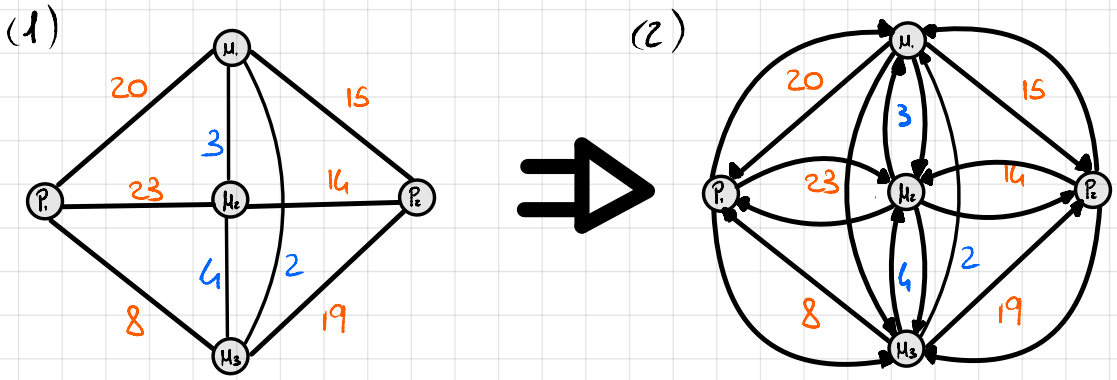
	M_1	M_2	M_3
P_1	20	23	8
P_2	15	16	19

COMMUNICATION COST

	M_1	M_2	M_3
M_1	0	3	2
M_2	3	0	4
M_3	2	4	0

- CUT formulation

(1) Un taglio sul seguente grafo rappresenta un assegnamento, il min $(P_1\text{-}P_2)$-CUT è la soluzione al problema.
(2) Per trovare il min CUT dobbiamo renderlo diretto



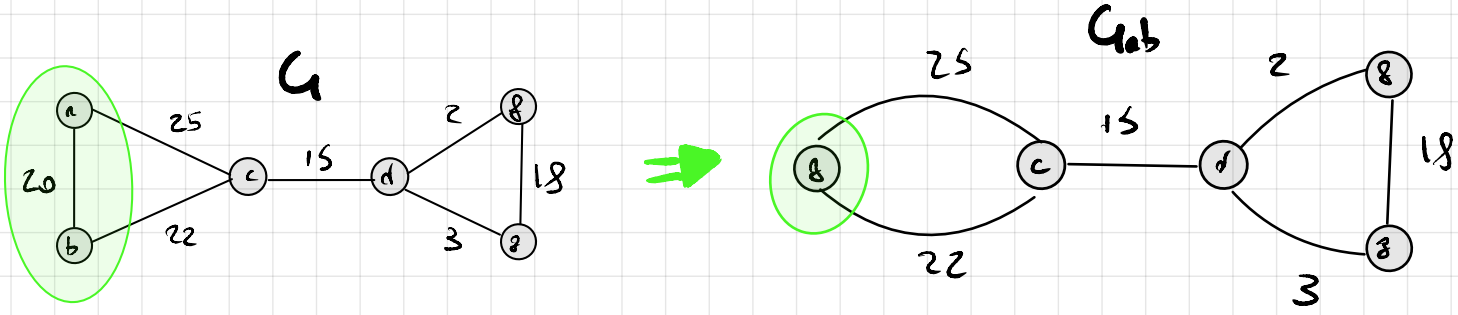
GLOBAL MIN CUT

- Per trovare un GLOBAL MIN CUT dobbiamo trovare i min cut per ogni coppia di nodi da separare
- Se il grafo è UNDIRECTED lo rendiamo diretto
- Le possibili coppie di nodi sono $n \cdot n$
- COMPLESSITÀ: $O(n^2) \cdot O[\text{MIN CUT}]$
- Possiamo ridurla NON eseguendo il min cut su coppie già viste $\left(\overset{\text{si}}{\blacksquare}(a,b) \in \overset{\text{No}}{(b,a)}\right)$

$\Rightarrow$ COMPLESSITÀ $O(n-1) \cdot \underbrace{O[\text{MIN CUT}]}$

$= O[\text{MAX FLOW MIN CUT}]$

NODE IDENTIFICATION

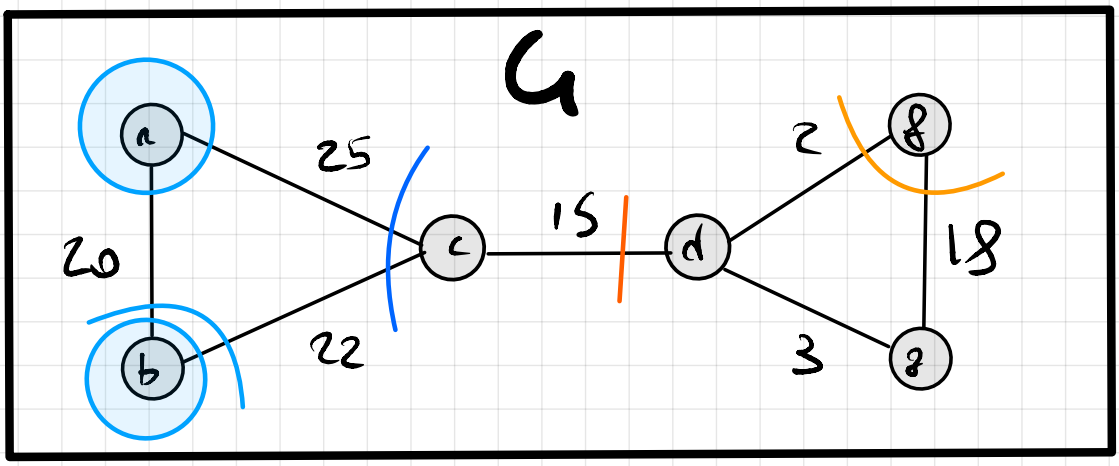


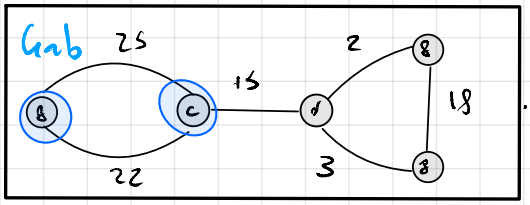
- Un CUT in G_{ab} è un cut anche in G
- Un CUT in G che non separa 'a' e 'b' è un CUT in G_{ab}

CONSEGUENZE

- sia $\lambda(G)$ il minimo cut di G
- sia $\lambda(G,a,b)$ il minimo cut di G che non separa 'a' e 'b'
- segue:

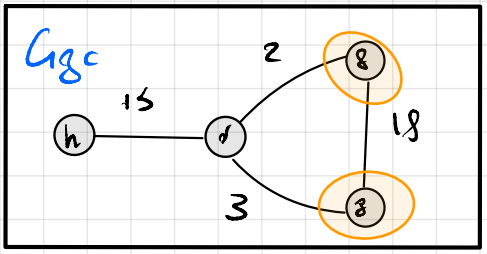
$$\lambda(G) = \min\{\lambda(G_{ab}), \lambda(G,a,b)\}$$





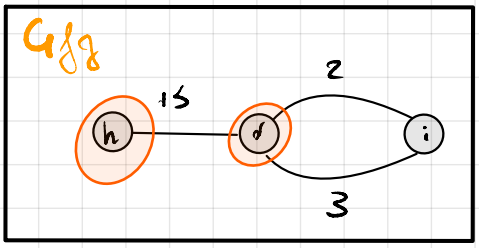
trovi amo $\lambda(G,a,b) = 42$ // F&F su G

$$\lambda(G) = \min\{\lambda(G_{ab}), 42\}$$



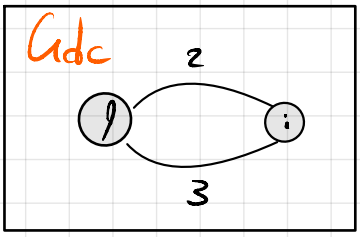
trovi amo $\lambda(G,g,c) = 47$ // F&F in G

$$\lambda(G) = \min\{\lambda(G_{gc}), 42, 47\}$$



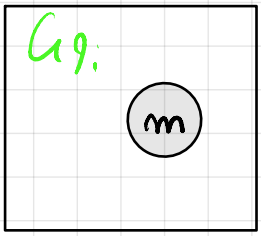
troviamo $\lambda(G,g,g) = 20$ // F&F su G

$\lambda(G) = \min\{\lambda(G_{gg}), 42, 47, 20\}$



troviamo $\lambda(G,d,h) = 15$ // F&F su G

$\lambda(G) = \min\{\lambda(G_{dh}), 42, 47, 20, 15\}$



troviamo $\lambda(G,i,i) = 5$ // F&F su G

$\lambda(G) = \min\{\lambda(G_{i:}), 42, 47, 20, 15, 5\}$

Algo

(1) scegli 2 nodi i, j
(2) trova il min cut tra i, j
(3) salva il valore del min cut tra i, j
identifica i, j e update G,
(4) Goto (1) finché i nodi di G sono più di due

} (n-1) esecuzioni

NOTA (2) è il Bottleneck

IDEA Scegliamo i nodi in (1) in modo tale da ridurre la complexity

(1) assegnare una LABEL $l_i = 0$ a ogni nodo

(2) Selezioniamo il nodo i con massima label (es 'a')

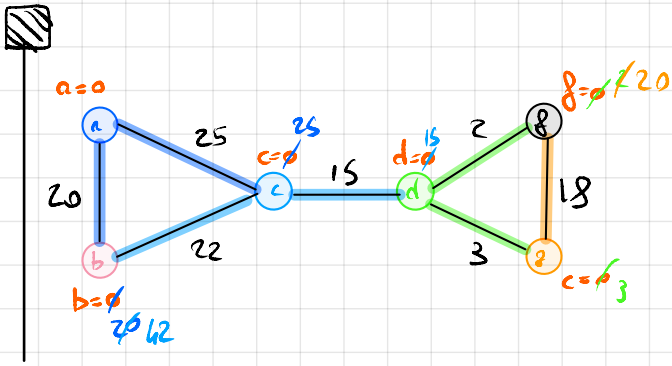
(3) Update delle label dei nodi vicini di i e non ancora selezionati

$l_j = l_j + c(i, v_k)$

(es b = 20 e c = 25)

} $O(n \cdot m)$ (n: nodi, m: viami)

(4) Goto (2)



Prendiamo i nodi nell'ordine

{a, c, b, d, g, &}

LEGAL ORDERING

$v_1, \dots, v_n$ è un LEGAL ORDERING dei vertici di G se dato $V_i = \{v_1, \dots, v_i\}$:

$$w(\delta(V_{i-1}) \cap \delta(V_i)) \geq w(\delta(V_{i-1}) \cap \delta(V_j))$$

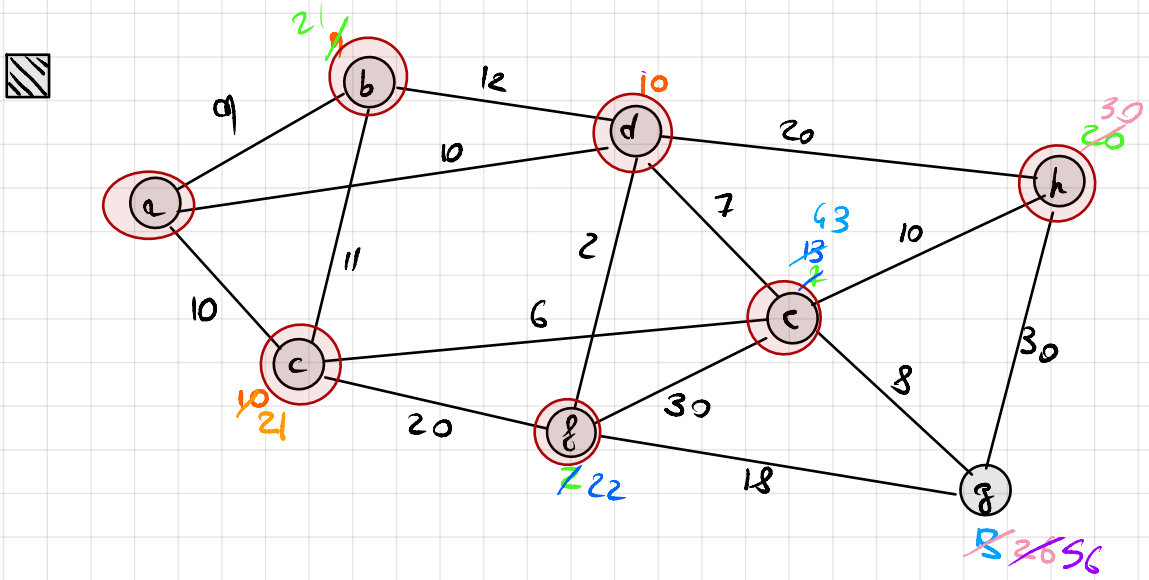
gli ultimi nodi dell'ordering $\{v_{n-1}, v_n\}$ li chiamo solito $\lambda(G, v_{n-1}, v_n) = \delta(v_n)$

L'ALGORITMO DIVENTA

(1) Calcola local ordering e prendi gli ultimi due $i, s = i-1$
(2) il valore del min cut tra i, j è $w(\delta(V_i))$
(3) salva il valore del min cut tra i, j
identifica i, j e update G,
(4) Goto (1) finché i nodi di G sono più di due

$O(n^3)$

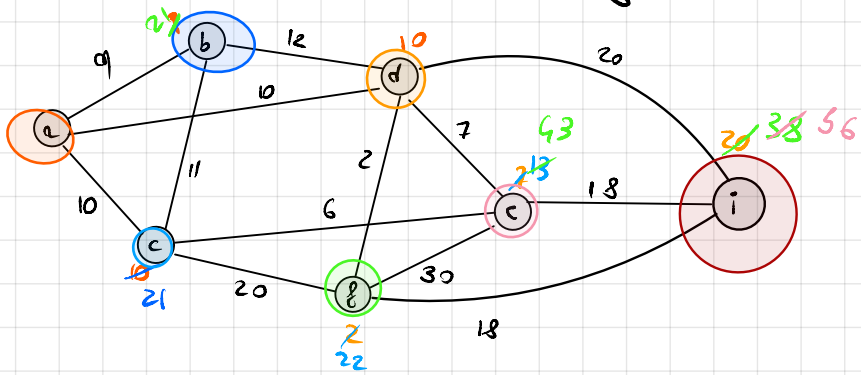
NOTA (1) è il Bottleneck ora



(1) LEGAL ORDERING $\{a, d, b, c, f, e, h, g\}$ → nodi (1)

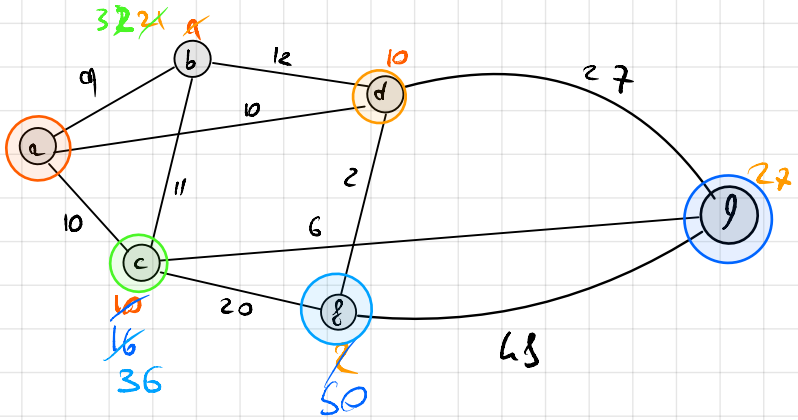
(2) $\rightarrow \lambda(G, h, g) = 56$

(3) IDENTIFICA 'g' e 'h' $\rightarrow G_{gh}$



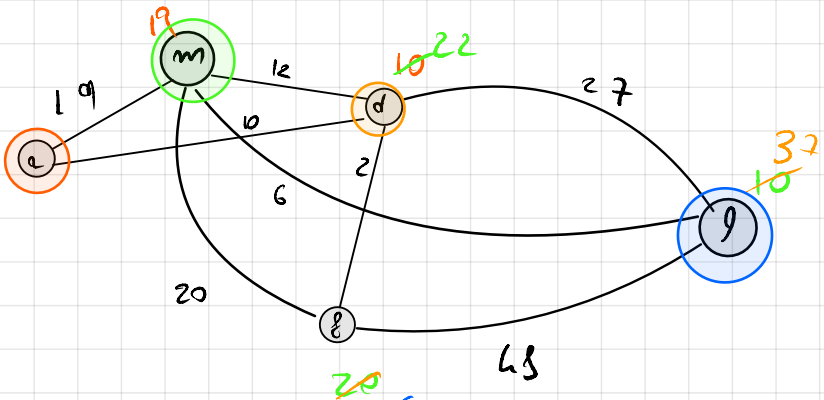
(1) $\{a, d, b, c, f, e, i\}$ (2) $\lambda(G, e, i) = 56$

③ IDENTIFICA 'c', 'i' ⇒ G_{ci}



① $\{a, d, 9, g, c, b\}$ ② $\lambda(G, b, c) = 32$

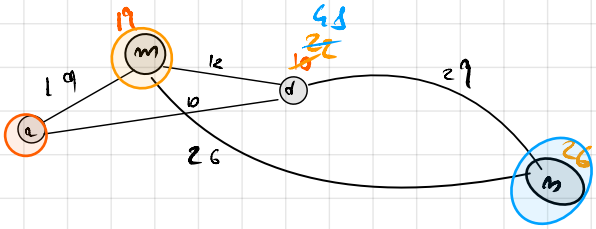
③ IDENTIFICA 'b', 'c' ⇒ G_{bc}



① $\{a, m, d, 9, g\}$

② $\lambda(G, b, c) = 70$

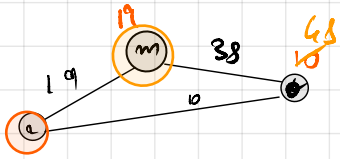
③ IDENTIFICA '9', 'g' ⇒



① $\{a, m, m, d\}$

② $\lambda(G, 9, g) = 49$

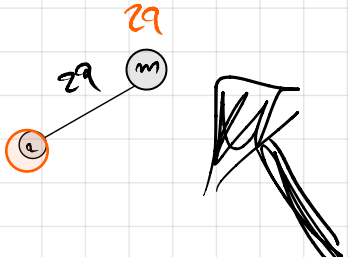
③ IDENTIFICA 'd', 'm' ⇒



① $\{a, m, \sigma\}$

② $\lambda(G, 1, m) = 49$

③ IDENTIFICA 'σ', 'm' ⇒

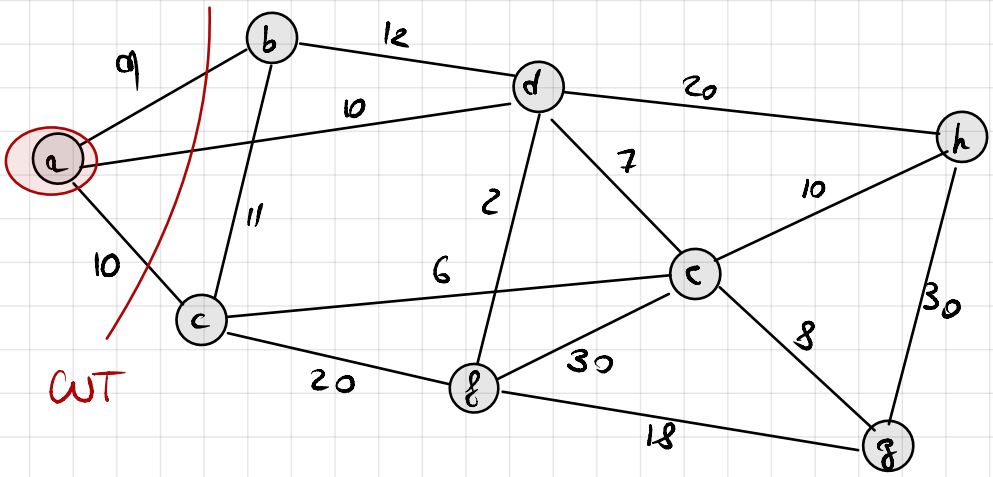


① $\{2, m\}$

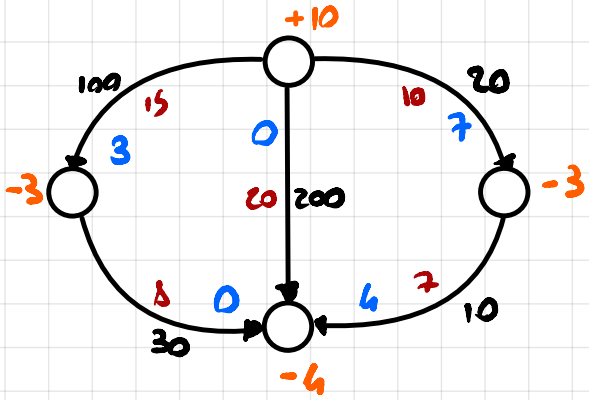
② $\lambda(G, \sigma, m) = 29$ min

⇒ $\lambda(G) = 29$

⇒ dobbiamo fare backtracking per trovare gli archi del taglio



MIN COST FLOW



$b(i)$

- Sia x_{ij} Flow su (i,j):

MIN COST FLOW PROBLEM

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

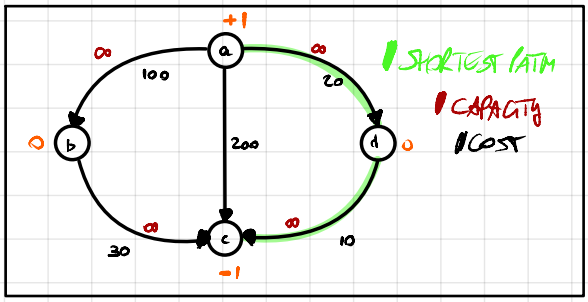
$$\sum_{(i,j)\in \delta(i)} x_{ij} - \sum_{(j,i)\in \delta(i)} x_{ji} = b(i) \qquad \forall i \in N$$

$$x_{ij} \geq 0$$

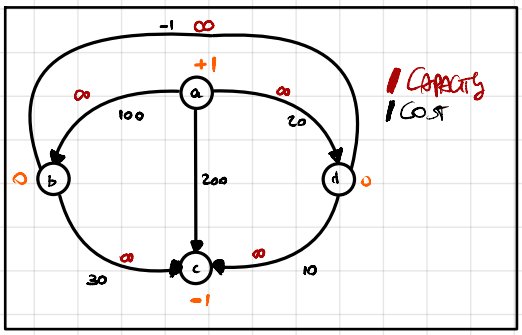
$$x_{ij} \leq u_{ij}$$

- MAX FLOW (MIN CUT)? SHORTEST PATH

- Settando i label su un nodo di partenza a +1 e su un nodo di arrivo a -1 (0 a tutti gli altri) risolvere il MIN COST FLOW equivale a risolvere lo Shortest path

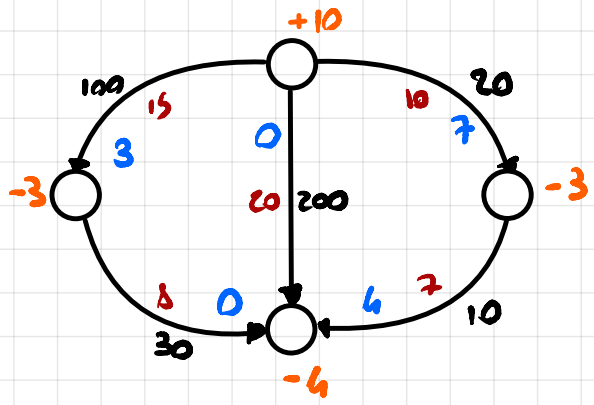


- Invece in un istanza con label a 0 e un arco extra tra nodo di partenza e uno di arrivo con costo -1



NODE-ARC INCIDENCE MATRIX (NETWORK MATRIX)

	ab	ac	ad	bc	dc	b
a	1	1	1			10
b	-1			1		-3
c		-1		-1	-1	-4
d			-1		1	-3



* RAPPRESENTA I COSTRAINT DI PRIMA

- IL DETERMINANTE può essere solo $\{0, 1, -1\}$
- Una soluzione ammissibile $X_B = B^{-1} b$ è sempre intera
- Per trovare una soluzione ci servono:
  - (1) · OPTIMALITY CONDITION
  - (2) · Trova una FEAS. SOLUTION
    - PIVOTING: meccanismo per spostarsi tra feasible solutions
    - Check di ottimalità sulla soluzione trovata
- Ci serve il duale del problema:

P

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

$$y_i: \sum_{(i,j)\in\delta(i)} x_{ij} - \sum_{(j,i)\in\delta(i)} x_{ji} = b(i) \quad \forall i \in N$$

$$z_{ij}: -x_{ij} \geq -u_{ij}$$

$$x_{ij} \geq 0$$

D

$$\max \sum_{i\in N} b_i g_i - \sum_{(i,j)\in A} u_{ij} z_{ij}$$

$$x_{ij}: \quad g_i - g_j - z_{ij} \leq c_{ij} \quad (i,j)\in A$$

$$g_i \leq 0$$

$$z_{ij} \geq 0$$

- Def: REDUCED COST $\bar{c}_{ij} = c_{ij} - g_i + g_j$
- WEAK DUALITY TH: If P has a feas. sol x^* and D too $g^*, z^* \Rightarrow cx^* = bg^* + uz^*$
- COMPLEMENTARY SLACKNESS if x feas. for P and y z feas. for D $\Rightarrow$ per g ottima y^*:

$$y^* \Big(\sum_{(i,j)\in\delta^+(i)} x_{ij}^* - \sum_{(j,i)\in\delta^-(i)} x_{ji}^* - b_i \Big) = 0$$

(PRIMO CONSTRAINT IN P)

$$z^*_{ij}\,(-x^*_{ij}+u_{ij})=0$$

$$x^*_{ij}\,(g_i-g_j-z_{ij}-c_{ij})=0$$

- TH: Optimality conditions min cost flow

if $\bar{c}_{ij}=c_{ij}-g_i+g_j<0$ then $x_{ij}=u_{ij}$

if $0<x_{ij}<u_{ij}$ then $\bar{c}_{ij}=0$

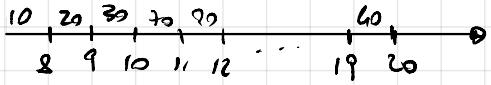
if $\bar{c}_{ij}>0$ then $x_{ij}=0$

WORK FORCE SCHEDULING

Contact center lavora dalle 8 alle 20

Part Time (PT) agent → 4 hours, 100 €/day

Full Time (FT) agent → 6 hours, 120 €/day



OBSERVE

Trova la configurazione di agenti che COPRA TUTTI GLI SLOT e abbia MINIMO COSTO

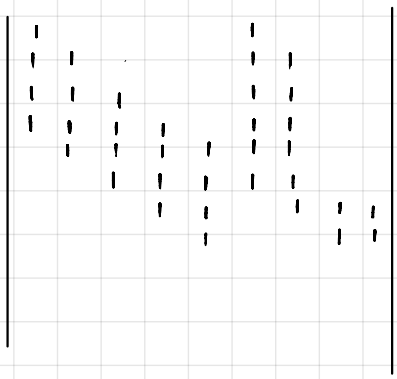
VARIABLES

x^{PT}_{start}, x^{FT}_{start} → #AGENTI che iniziano al tempo START

LAVORANO 6 ORE

$$\begin{aligned}
&x_1^{PT} && && && && && \le 10\\
&x_1^{PT}+x_2^{PT} && && && && \le 20\\
&x_1^{PT}+x_2^{PT}+x_3^{PT} && && && && \le 30\\
&x_1^{PT}+x_2^{PT}+x_3^{PT}+x_4^{PT} && && && && \le 70\\
&x_2^{PT}+x_3^{PT}+x_4^{PT}+x_5^{PT} && +\ x_1^{FT} && && && \le 90\\
&x_3^{PT}+x_4^{PT}+x_5^{PT} && +\ x_1^{FT}+x_2^{FT} && && && \le 50\\
&x_4^{PT}+x_5^{PT} && +\ x_1^{FT}+x_2^{FT}+x_3^{FT} && && && \le 20\\
&x_6^{PT} && +\ x_1^{FT}+x_2^{FT}+x_3^{FT}+x_4^{FT} && +\ x_5^{FT} && && \le 5
\end{aligned}$$

CONSECUTIVE ONE PROPERTY



- Dato un problema:

$$\begin{aligned} \min \quad & cx \\ & Ax \geq b \\ & x \geq 0 \end{aligned}$$

← A ha CONSECUTIVE 1 PROPERTY

- Lo trasformiamo nella forma Standard $Ax \geq b \Rightarrow Ax - Iy = b$

- Aggiungo una riga di 0
- Sottraggo ogni riga con quella sopra e ricavo il problema

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & y_1 & y_2 & y_3 & y_4 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 4 \\ -7 \\ -7 \end{bmatrix}$$

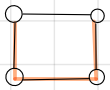
NETWORK SIMPLEX ALGORITHM

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

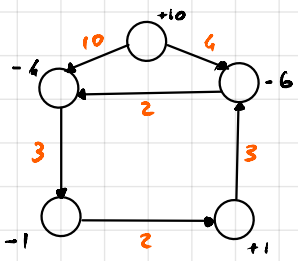
s.t.

$$\sum_{(i,j)\in \delta^+(i)} x_{ij} - \sum_{(j,i)\in \delta^-(i)} x_{ji} = b(i) \quad i \in N$$

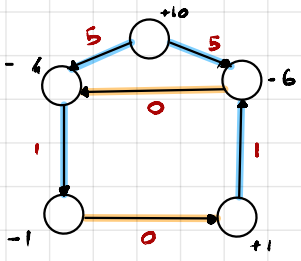
$$x_{ij}$$

- se G è connesso $\Rightarrow$ G ammette uno spanning tree T ()
- Una feasible solution x per P è detta TREE SOLUTION, se dato uno spanning tree T è T/C:

$$\begin{cases} \sum_{(i,j)\in \delta^+(i)} x_{ij} - \sum_{(j,i)\in \delta^-(i)} x_{ji} = b(i) & i \in N \\ x_{ij} = 0 \quad \forall (i,j) \notin T & \text{// tutte le vars fuori da } T \text{ sono } 0 \end{cases}$$



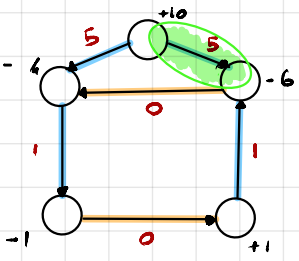
- c_{ij}
- $u_{ij} = \infty$



- feasib. x_{ij}
- Arcs | $x_{ij} = 0$
- Arcs | $x_{ij} > 0$ $\Rightarrow T$

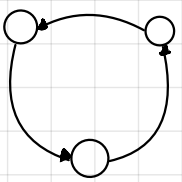
$\Rightarrow$ We have a TREE SOLUTION T

$\Rightarrow$ The value of the solution is $10 \cdot 5 + 4 \cdot 5 + 3 \cdot 1 + 3 \cdot 1 = 76$



NOTA che preso un arco può partizionare il grafo in modo che il flow che va da $PART_1$ a $PART_2$ è portato da esattamente un ARCO

LEMMA Dato un tree T, esiste sempre un unica TREE SOLUTION associata a T



FTS $\rightarrow$ x è determinata univocamente

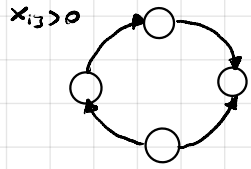
$X \rightarrow$ porta a più FST

TH se P ammette una feas. solution allora ammette una FTSolution

se P " " optimal solution " " un OTTIMA FTSolution

Nota Possiamo determinare un algoritmo che si sposta solo tra FTSolution fino a trovare l'ottimo

PROOF Sia X feas. per P ma non FTSol.

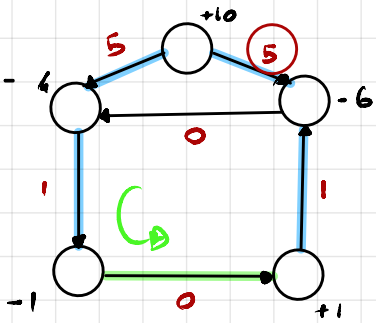


$\varepsilon = \min x_{ij}$ tra i REVERSE ARCS

$x'_{is} = x_{is} - \varepsilon$ se (is) REVERSE

$x'_{ij} = x_{ij} + \varepsilon$ se (ij) FORWARD

MUOVERSI TRA FTS



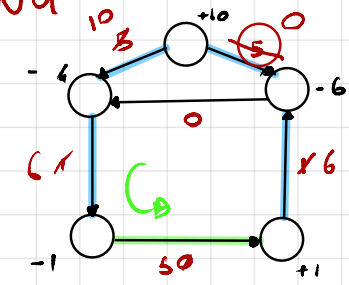
- Partendo da FTSol
- Come ci spostiamo a un altra FTSol
- Aggiungo un arco a FTS

↳ ora ho un ciclo, e l'arco aggiunto è orientato G

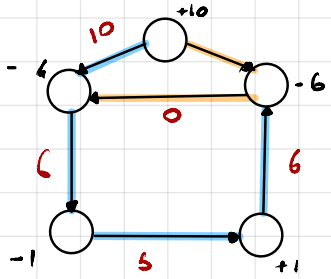
↳ tra ogni arco nel ciclo scelgo quello orientato verso (2) con $\min x_{ij}$

↳ incremento il flow di ◯ agli archi verso 'G' e lo sottraggo a quelli verso 'D'

PIVOTING



⇒

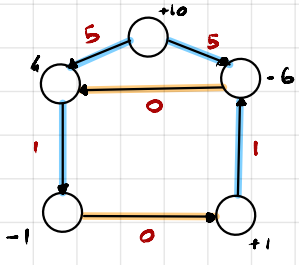


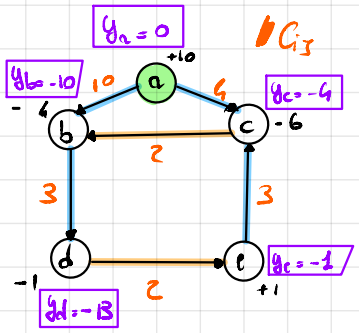
T Nuovo FTSol

L archi con $x_{ij} \neq 0 \mid (ij) \notin T$

- Come checkiamo l'ottimalità?

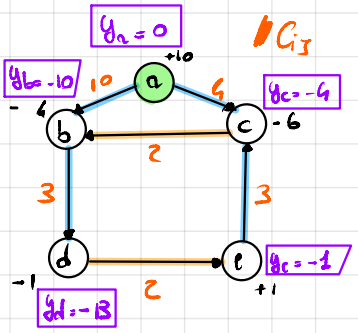
- Rc. REDUCED COST : $\bar{c}_{ij} = c_{ij} - y_i + y_j$





- Presa FTSol
- Scegliamo ROOT
- Troviamo $y_h = -\sum_{(ij)\ \text{forward}} c_{ij} + \sum_{(ij)\ \text{Reverse}} c_{ij}$ sul path $r \to h$ su T

• Per costruzione $\bar{C}_{ij}$ sugli archi in T sono tutti ϕ



$C_{cb} = 2+4-10 = -4$

$C_{de} = 2+13-1 = 14$

∞∞∞

RIC

• L'optimality condition è:

• TH: Optimality condition min cost flow

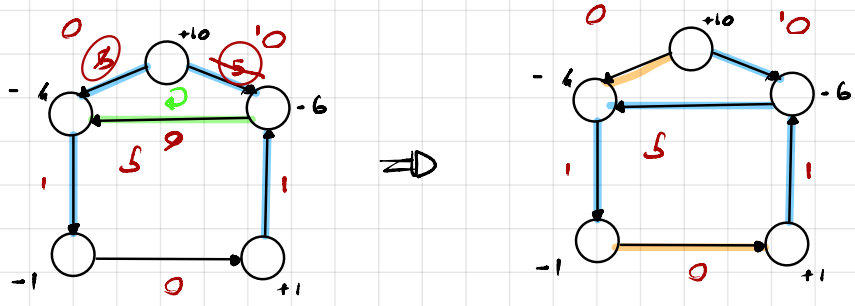
① If $\bar{C}_{ij} = C_{ij} - g_i + g_j < 0$ then $x_{ij} = l_{ij}$

② if $0 < x_{ij} < l_{ij}$ then $\bar{C}_{ij} = 0$

③ if $\bar{C}_{ij} > 0$ then $x_{ij} = 0$

∞∞ la soluzione non è ottima: ① $C_{cb} < 0$ quindi aggiungo l'arco nel TREE poiché setto $x_{cb} = l_{cb}$

Ora lo devo togliere con il PIVOT



NEW TREE!!

• Continuo finché non soddisfi OPT CONDITION

NETWORK SIMPLEX ALGORITHM

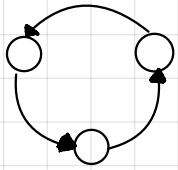
$u_{ij} = +\infty$

Trova una FTSol. T, L

Calcola i potenziali y per ogni nodo tramite una corrispondenza i

setta $y_i = 0$

WHILE $(\exists (i,j) \in L$ t.c. $\bar{c}_{ij} < 0)$



Sia c il ciclo ottenuto aggiungendo (i,j) a T

Sia C orientato come (i,j)

IF (C non ha un REVERSE ARC)

STOP

ELSE

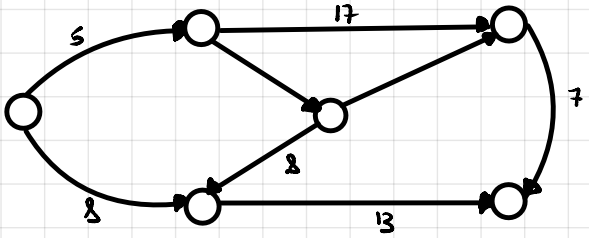
Sia $\varepsilon = \min \{ x_{hk}$ t.c. (h,k) è reverse in $C \}$

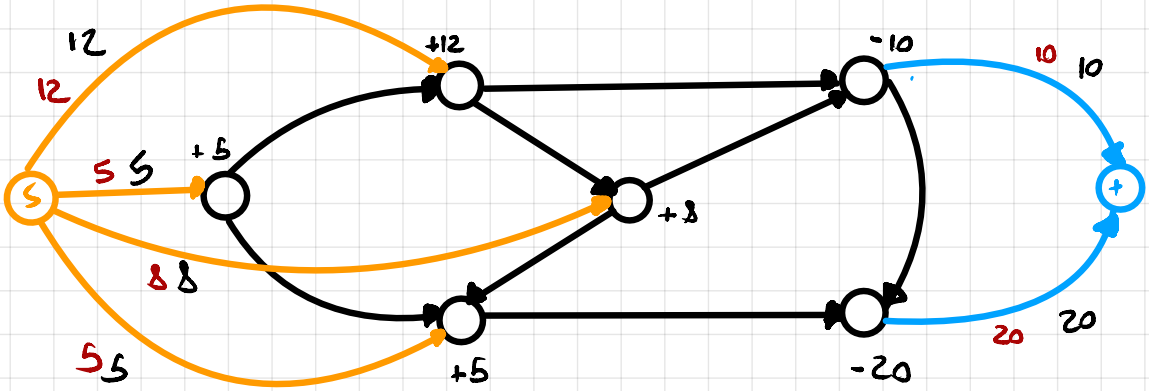
Seleziona un REVERSE ARC (l,m) | $x_{lm} = \varepsilon$

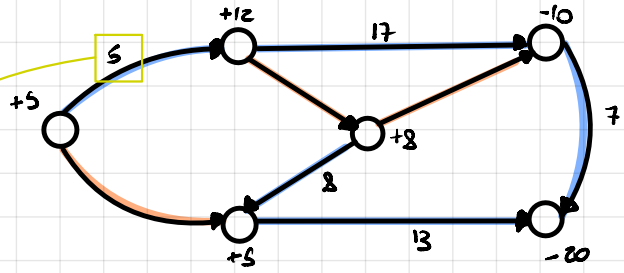
Cambia il flow di ε sul ciclo C

$T = T / \{(h,k)$ t.c. $x_{hk} = 0\}$

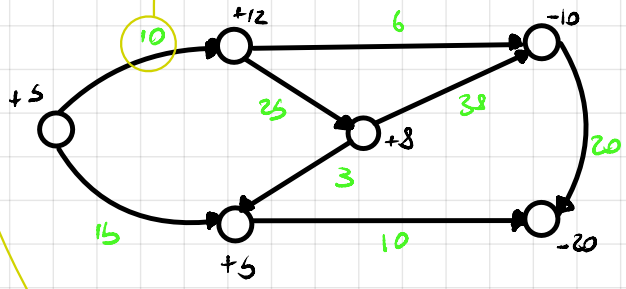
Update y



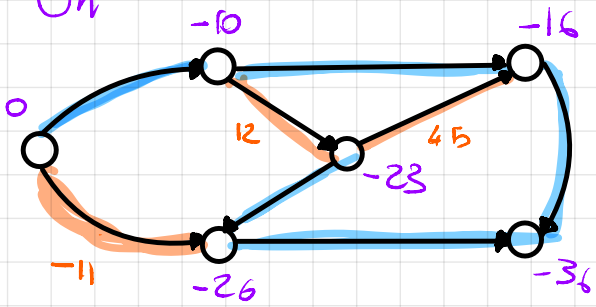




Costi

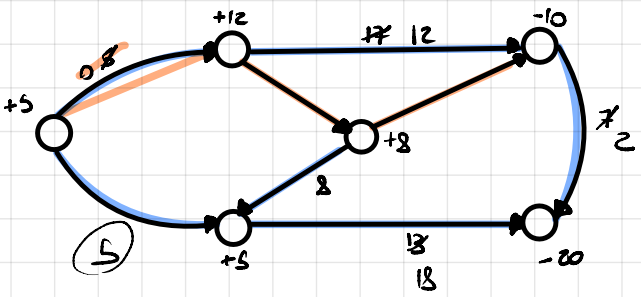


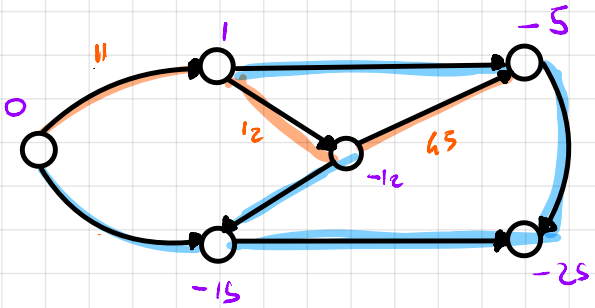
y_h $\bar{c}_{is}$



$z = 5 \cdot 10 + 17 \cdot 6 + 7 \cdot 20 + 13 \cdot 10 + 8 \cdot 3 = 446$

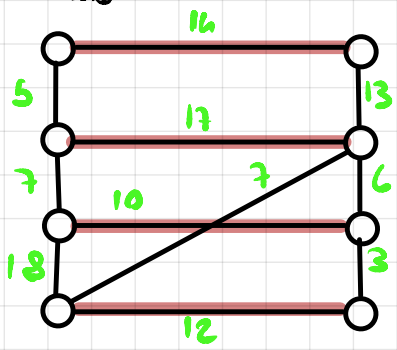
- Ora aggiungiamo un ARCO $\sim T$





- Sol ottima → i costi ridotti di $\mathcal{L}$ sono tutti ≥ 0

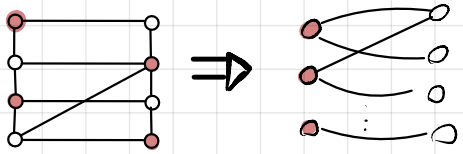
- Dato



Perfect matching → $z = 16 + 17 + 10 + 12$

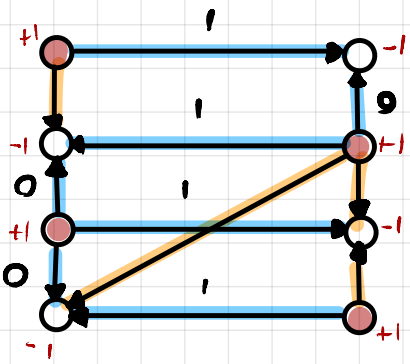
Costs

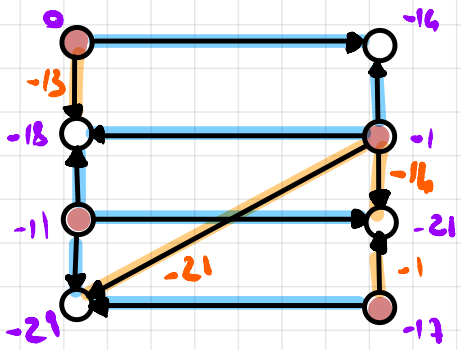
NOTA: G è BIPARTITO

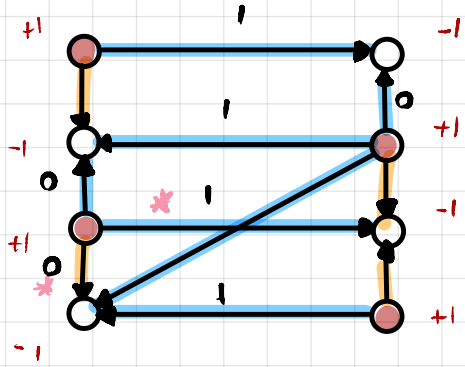


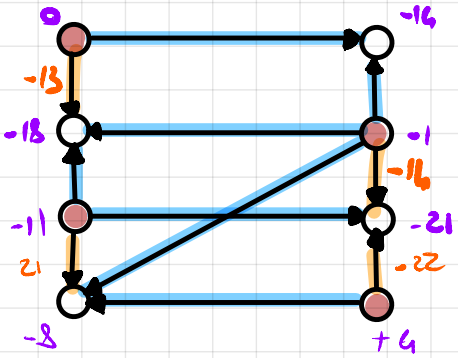
- Trova il PERF. MATCH. di minimo costo

$\{T, L\}$



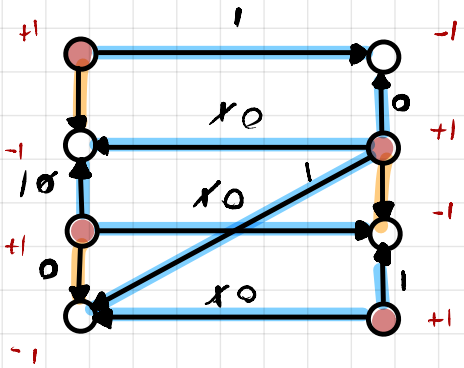


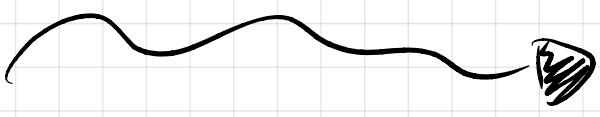




*DEGENERATE ITERATION

NOTA: AGGIUNGENDO L'ARCO CON -21, CA È SUL CICLO È Ø, QUINDI DEVO SPOSTARE L'ARCO CON Ø IN L





NOTA, ci sono 3 archi con $x_{ij} = 0$, ne dobbiamo togliere solo 1

☐ Valuta opt. del PL seguente

x_6 non è nei constr. quindi, cercando il minimo, $x_6^* = 0$ sempre

$$\min \quad 51x_1 + 34x_2 + 27x_3 + 16x_4 + 25x_5 + 19x_6$$

$$\text{s.t.}$$

$$x_1 + x_4 \geq 25$$

$$x_2 + x_3 + x_5 \geq 23$$

$$x_1 + x_2 + x_3 + x_5 \geq 21$$

$$x_1 + x_2 + x_4 + x_5 \geq 39$$

$$x \geq 0$$

LA MATRICE HA LA CONSECUTIVE 1 PROPERTY?

	x_1	x_2	x_3	x_4	x_5
a	1	0	0	1	0
b	0	1	1	0	1
c	1	1	1	0	1
d	0	1	0	0	1
e	1	1	0	1	1

SWAP

	x_1	x_2	x_3	x_4	x_5
a	1	0	0	1	0
e	1	1	0	1	1
c	1	1	1	0	1
d	0	1	0	0	1
b	0	1	1	0	1

SWAP

	x_1	x_2	x_3	x_4	x_5
a	1	0	0	1	0
e	1	1	0	1	1
c	1	1	1	0	1
b	0	1	1	0	1
d	0	1	0	0	1

- Ha 9a C.1 PROP! TRASFORMIAMOCA IN UNA NETWORK MATRIX

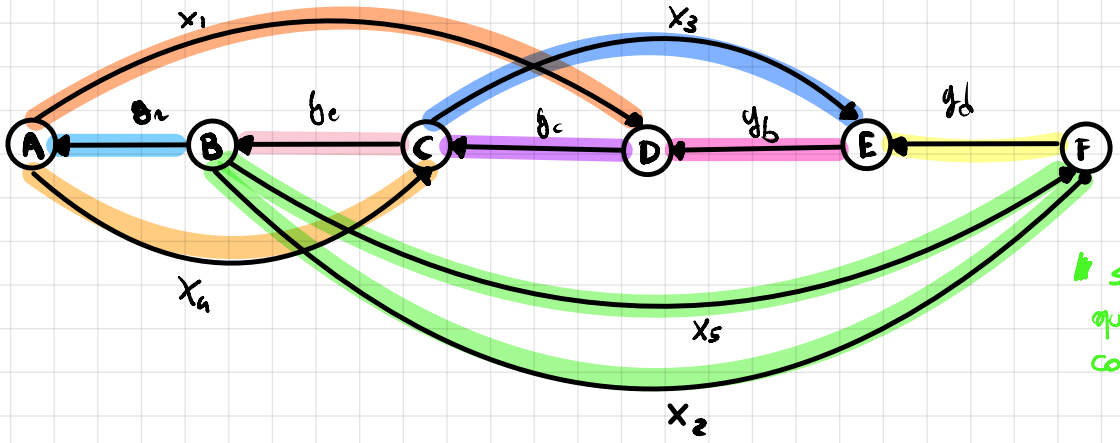
	x_1	x_2	x_3	x_4	x_5
a	1	0	0	1	0
e	1	1	0	1	1
c	1	1	1	0	1
b	0	1	1	0	1
d	0	1	0	0	1

	x_1	x_2	x_3	x_4	x_5	y_a	y_e	y_c	y_b	y_d	RHS
a	1	0	0	1	0	-1					25
e	1	1	0	1	1		-1				39
c	1	1	1	0	1			-1			21
b	0	1	1	0	1				-1		23
d	0	1	0	0	1					-1	14
δ	0	0	0	0	0	0	0	0	0	0	0

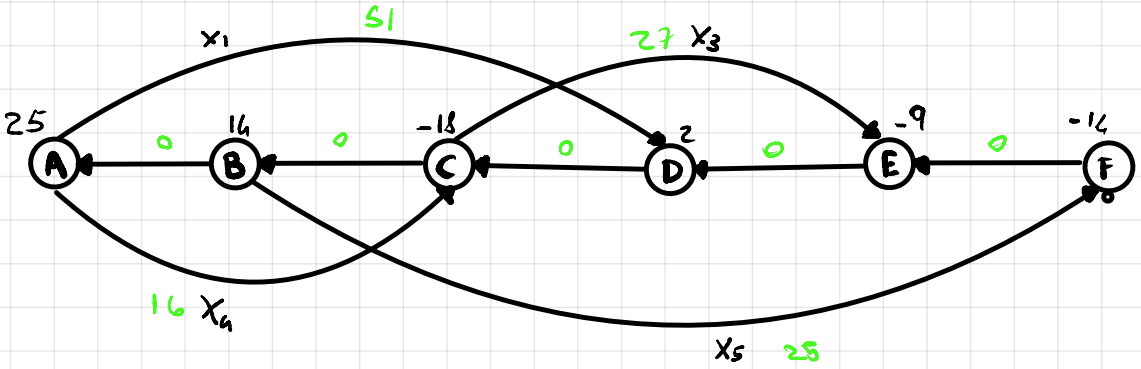
NODES \ ARCS	x_1	x_2	x_3	x_4	x_5	y_a	y_e	y_c	y_b	y_d	RHS
A = a	1	0	0	1	0	-1	0	0	0	0	25
B = e-a	0	1	0	0	1	1	-1	0	0	0	14
C = c-e	0	0	1	-1	0	0	1	-1	0	0	-18
D = b-c	-1	0	0	0	0	0	0	1	-1	0	2
E = d-b	0	0	-1	0	0	0	0	0	1	-1	-9
F = δ-d	0	-1	0	0	-1	0	0	0	0	1	-14

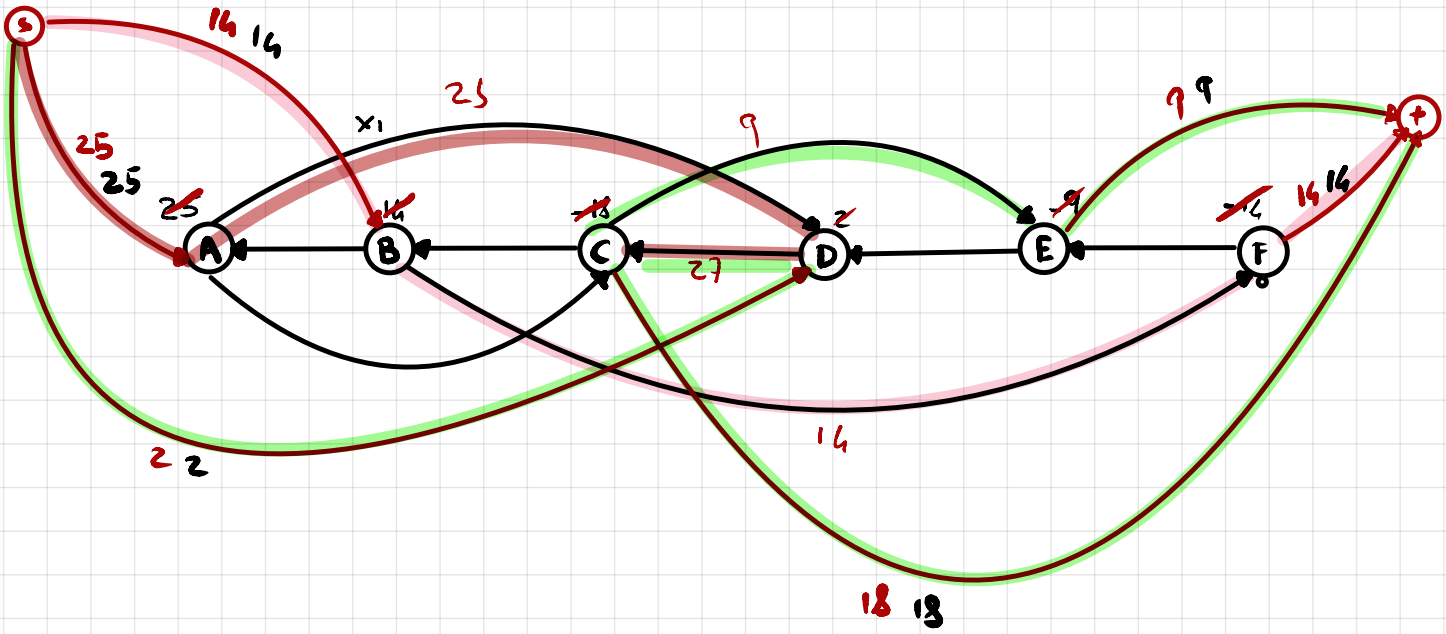
- È UNA NETWORK MATRIX?
  - LA TRASFORMAZIONE È CORRETTA PERCHÉ OGNI COLONNA È UN 1 E UN -1
  - LA SOMMA DEGLI RHS DA 0

NODES \ ARCS	x_1	x_2	x_3	x_4	x_5	y_a	y_c	y_c	y_b	y_d	RHS
A	1	0	0	1	0	-1	0	0	0	0	25
B	0	1	0	0	1	1	-1	0	0	0	14
C	0	0	1	-1	0	0	1	-1	0	0	-18
D	-1	0	0	0	0	0	0	1	-1	0	2
E	0	0	-1	0	0	0	0	0	1	-1	-9
F	0	-1	0	0	-1	0	0	0	0	1	-14

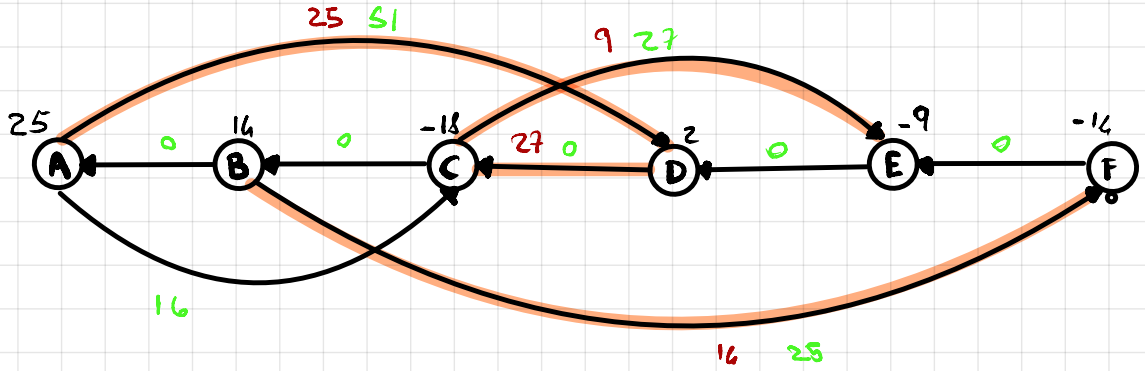


scelgo di tenere quello con minimo costo

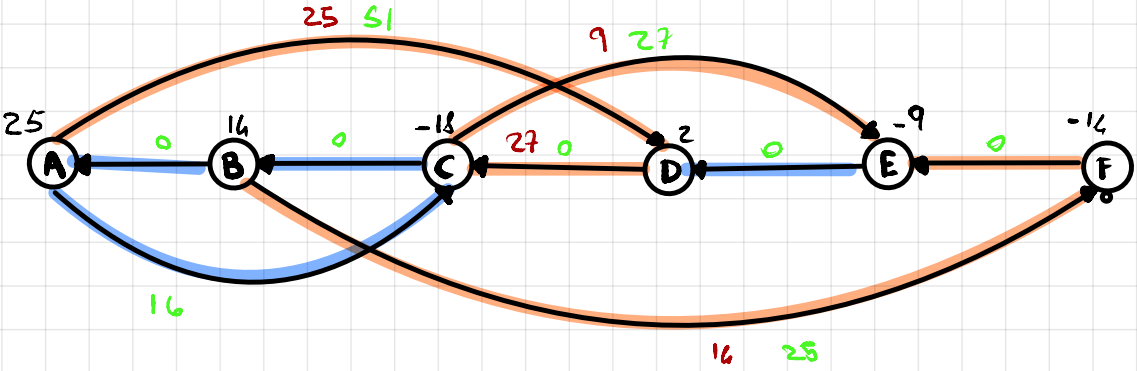




- Trova FEAS. T, L SOLUTION



- Dobbiamo rendere T un albero



- $Z = 25 \cdot 51 + 16 \cdot \quad + 9 \cdot 27 = 1868$

0

0

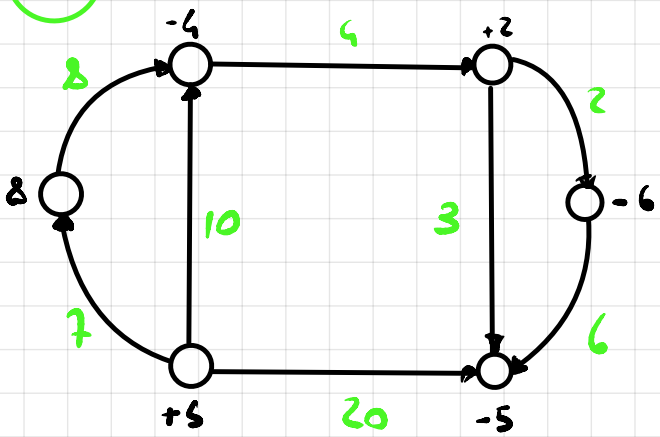
0

NETWORK SIMPLEX CON $u_{ij} \neq +\infty$

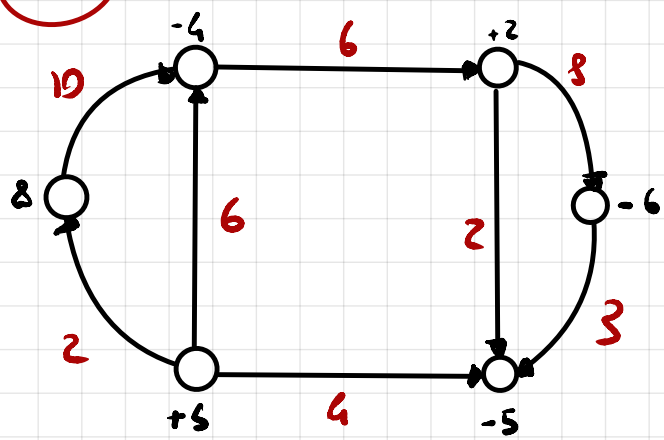
UN FTS T, L, U è feasible per P se

$$\begin{cases} \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(j,i)\in\delta^-(i)} x_{ji} = b(i), & i \in N \\ x_{ij} = 0 & \forall (i,j) \in L \\ x_{ij} = u_{ij} & \forall (i,j) \in U \end{cases}$$

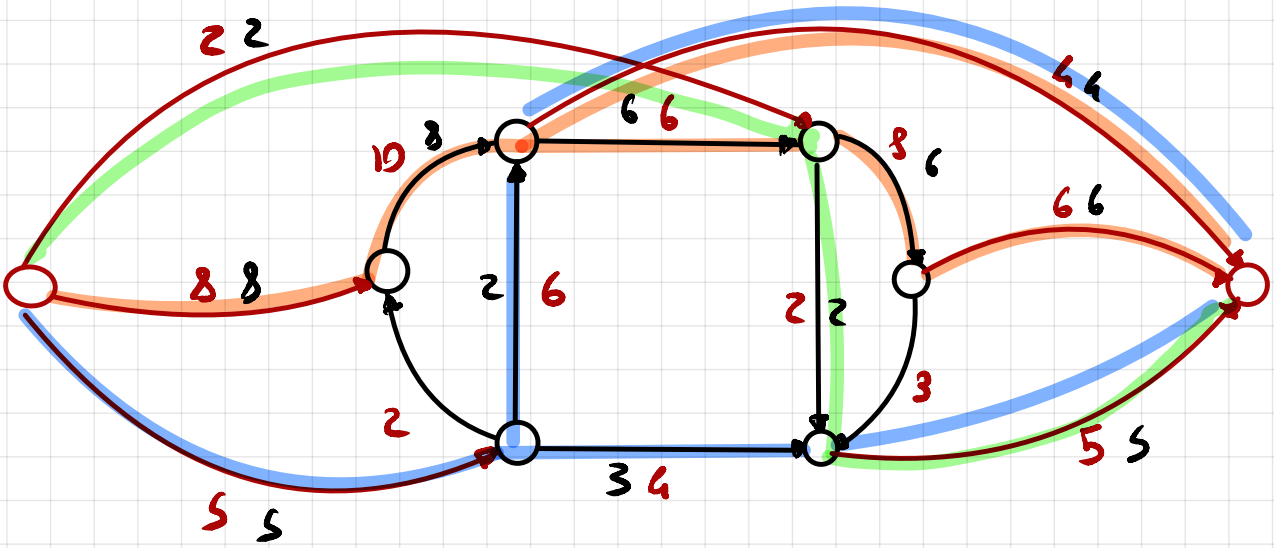
c_{ij}



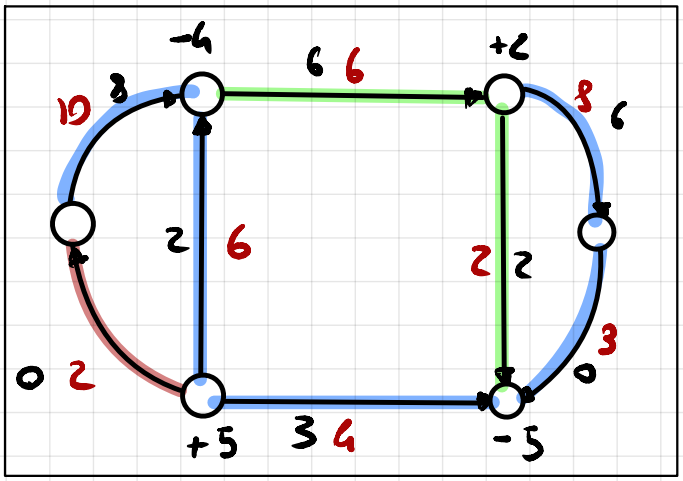
u_{ij}



- Cerco un feasible flow per poi trovare T, L, U



T L U

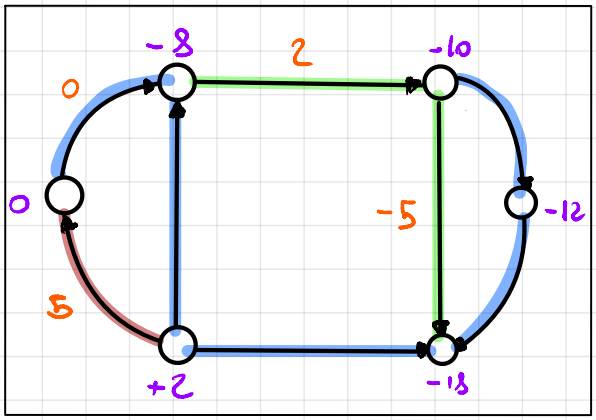


- Gli archi T/C $0 < x_{ij} < l_{ij}$ vanno in T
- Gli archi SATURI vanno in U
- Se T non è un tree, aggiungiamo archi con $x_{ij}=0$ per farcelo diventare

- Troviamo $z = \sum \text{COSTO} \cdot \text{FLOW}$ sugli archi con $x_{ij}>0$ (Quindi $T \cup U$)

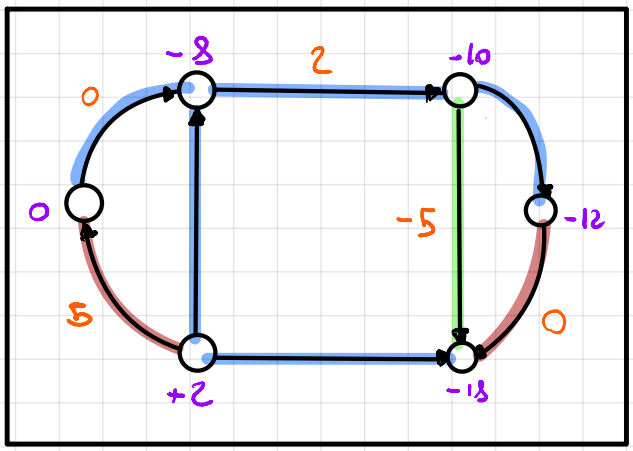
$$\Rightarrow z = 8\cdot 8 + 2\cdot 10 + 6\cdot 6 + 20\cdot 3 + 2\cdot 3 + 6\cdot 2 = 186$$

- Trova i reduced cost $\bar{c}_{ij}$ (disegnamo il grafo con il proprio POTENZIALE y)



- Per avere OPT:
  - $\bar{c}_{ij} \geq 0$ per L
  - $\bar{c}_{ij} \leq 0$ per U

- Aggiungendo l'arco verde otteniamo una DEGENERATE ITERATION, quindi rimuoviamo l'arco blu con $x_{ij}=0$



NETWORK SIMPLEX ALGO PER $u_{ij} \neq \infty$

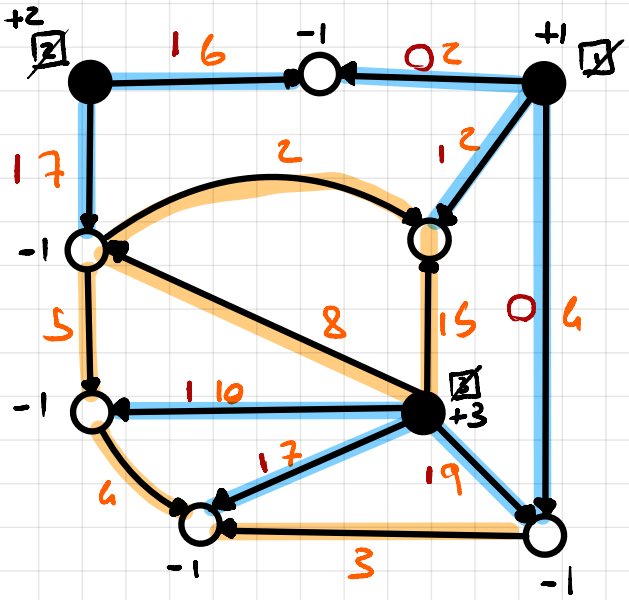
1. Trova FTS $\{T, L, U\}$
2. Calcola Potenziali y
3. WHILE ($\exists (i,j) \in L \cup U$ t/c $\bar{c}_{ij} = c_{ij} - y_i + y_j \gtrless 0$ OR
   - Aggiungi (i,j) a T
   - IF $((i,j) \in L)$
     - Orienta il ciclo C nel verso di (i,j)
   - ELSE IF $((i,j) \in U)$
     - Orienta il ciclo C nel verso opposto di (i,j)
   - IF (C non contiene REVERSE o FORWARD di capacità finita)
     - STOP
   - $\varepsilon_1 = \min \{ x_{ij} \mid (i,j)$ REVERSE rispetto $C \}$
   - $\varepsilon_2 = \min \{ u_{ij} - x_{ij} \mid (i,j)$ FORWARD rispetto $C \}$
   - $\varepsilon = \min \{\varepsilon_1, \varepsilon_2\}$
   - Scelgo un arco con $x_{hk} = 0$ OR $x_{hk} = u_{hk} \to (l,m)$
   - $T = T \cup (i,j) \setminus (l,m)$
   - IF $x_{lm} = 0$ $L = L \cup (l,m)$
   - IF $x_{lm} = u_{lm}$ $U = U \cup (l,m)$
   - UPDATE POTENTIALS y

Exercise 3 (27 MAGGIO 2019)

The following graph represents a distribution network in which black nodes are warehouses and white nodes customers. Each customer requires one unit of a good that can be supplied from any warehouse. Figures on the arcs represent the cost of shipping the good on that arc (arcs are uncapacitated), while numbers in the box represent the availability of the good in each warehouse.

1. Find the minimum cost shipping solution.

[Bonus] 2. Suppose to have the possibility of moving all goods from the central warehouse to the warehouse on the top right corner. How much would you be willing to pay for this?

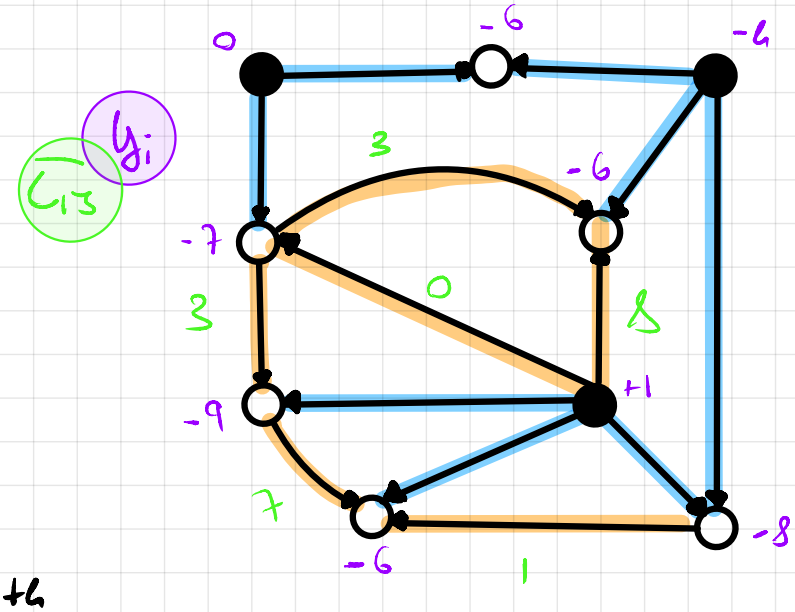


× WHARE HOUSES CAPACITY

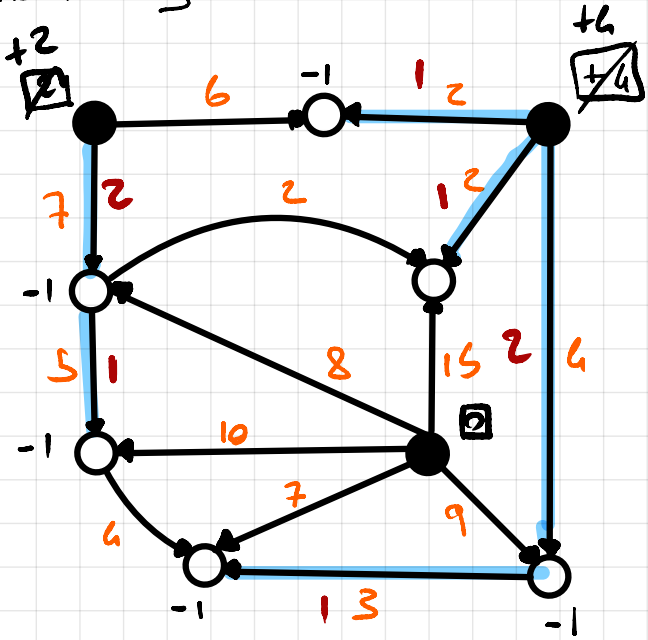
C_{ij}, X_{ij}

{ T L } // Ce lo troviamo senza il metodo che aggiunge set + (BASTA UN FEAS FLOW X)

$Z = 6+7+2+10+7+9 = 41$



[BONUS]



[Bonus] 2. Suppose to have the possibility of moving all goods from the central warehouse to the warehouse on the top right corner. How much would you be willing to pay for this?

- Da qui non si va avanti, se z è migliore allora DAJE, sennò non conviene

$Z = 14+5+2+2+8+3 = 34$