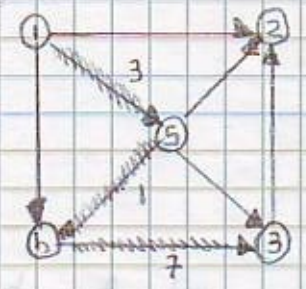
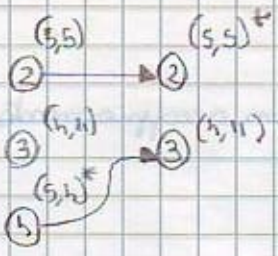
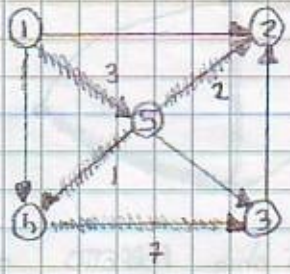
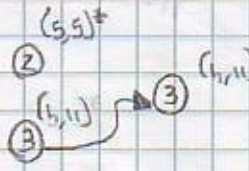






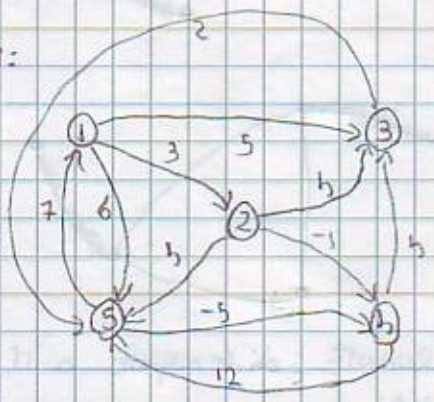
Quindi:



L'algoritmo di Dijkstra purtroppo non funziona quando gli archi hanno costi anche negativi. Esiste un algoritmo che permette di aggirare questo problema, se si sa a priori che G non contiene cicli di lunghezza totale negativa. La complessità di tale algoritmo è $O(n^3)$ e si chiama ALGORITMO DI FORD. Per esempio:

	1	2	3	4	5
1	0	3	5	∞	6
2	∞	0	4	-1	4
3	∞	∞	0	∞	2
4	∞	∞	4	0	12
5	7	∞	1	-5	0

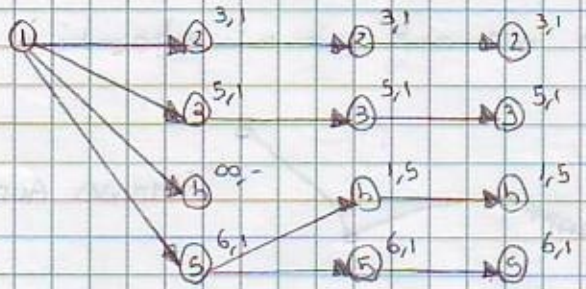
GRAFO:



L'idea dell'algoritmo consiste nell'esplorare il grafo e determinare i percorsi dall'origine s ad ogni altro nodo del grafo considerando via via il migliore finora trovato. Usa quindi una doppia etichetta:

$1° \rightarrow \lambda_j$: lunghezza percorso da s a j
$2° \rightarrow p_j$: nodo precedente a j.

Quindi:



Si noti che nella II° iterazione l'arco da 1 a 3 è più economico dell'arco da 2 a 3. Inoltre l'arco (3,4) consente di raggiungere il nodo 4 in maniera più economica. Consideriamo ora

Sia $\Pi = \dots$ un sottoinsieme dei lati di G.

Un ACCOPPIAMENTO è un sottoinsieme Π di lati di G tale che ogni vertice V di G è incidente al più ad un lato di Π.

Per esempio:

non è un accoppiamento.

Un accoppiamento si dice PERFETTO quando ogni vertice di G appartiene ad esattamente un lato di Π. Per esempio:

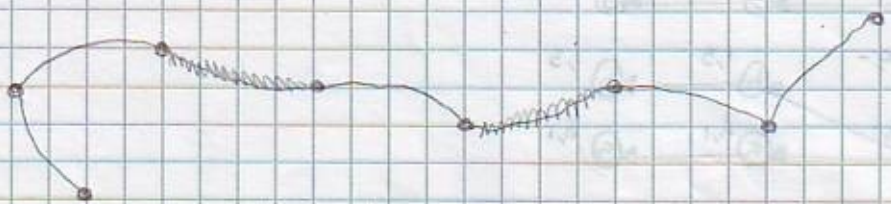
ACCOPPIAMENTO PERFETTO

ACCOPPIAMENTO NON PERFETTO.

Si dice CAMMINO ALTERNANTE di G rispetto a Π un cammino dove tutti i lati $(v_1, v_2), (v_3, v_4)$ sono in Π. Per esempio:

CAMMINO ALTERNANTE.

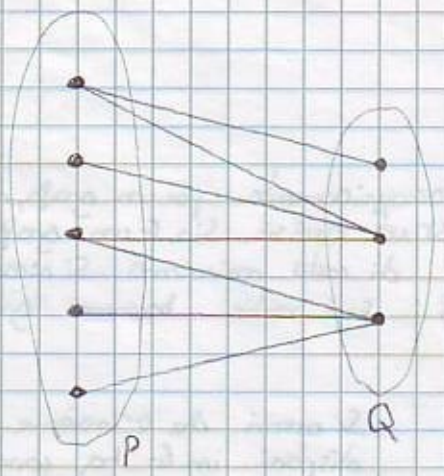
Invece un CAMMINO AUMENTANTE è un cammino alternante in cui entrambi i vertici terminali sono liberi. Per esempio:



CAMMINO AUMENTANTE.

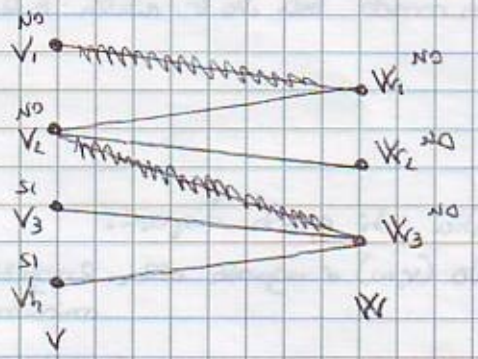
Per aumentare un cammino alternante posso eliminare i lati di Π inserisci lati non Π, e dove ci sono i lati non Π metteci lati di Π. Per esempio:

Ma per capire che Π ha cardinalità massima, dobbiamo trovare il cammino aumentante P. Esiste un algoritmo che ci permette di trovare un cammino aumentante P sia sui grafi bipartiti e quindi non funziona sui grafi qualunque. Tale algoritmo ha complessità polinomiale. Si ricordi che un GRAFO BIPARTITO è un grafo del seguente tipo:



Sia G un grafo bipartito e siano $V \in W$ le due sporte di vertici ed E l'insieme dei lati.

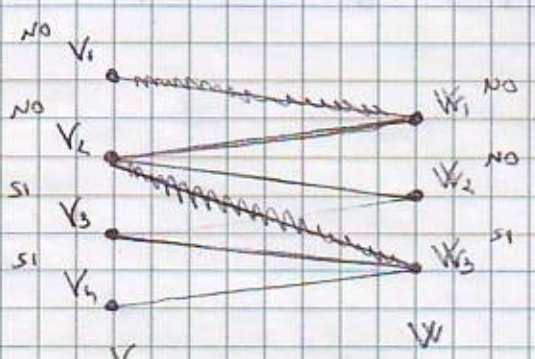
π = lati di Π.



1) etichettiamo con NO tutti i vertici di W e tutti i vertici di V incidenti a lati di π. Etichettiamo invece con SI tutti i vertici di V non occupati. Siamo:

$S = (V_3, V_4)$

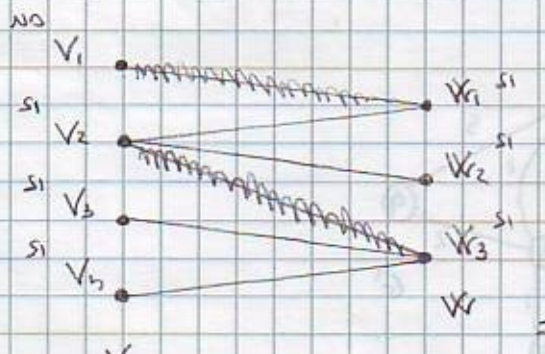
$T = \emptyset$ → insieme



2) Etichettiamo con SI tutti i vertici di W aventi etichetta NO, che però sono connessi ad un vertice v di S tramite un lato non appartenente ad Π.

$T = \{W_3\}$

$S = \emptyset$



3) Se qualche w_i di T non è accoppiato abbiamo trovato P, altrimenti etichettiamo tutti i vertici di V aventi etichetta NO connessi ad un w di T con un lato di Π, con un SI. Quindi:

$S = \{V_4\}$

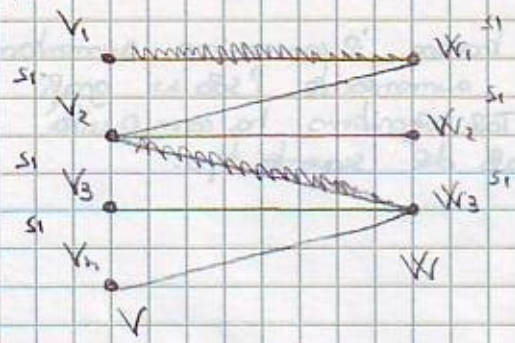
$T = \{\emptyset\}$

4) STOP, se P è stato trovato o se $S = \emptyset$, altrimenti si torna al passo 2.

→ $T = \{W_1, W_2\}$

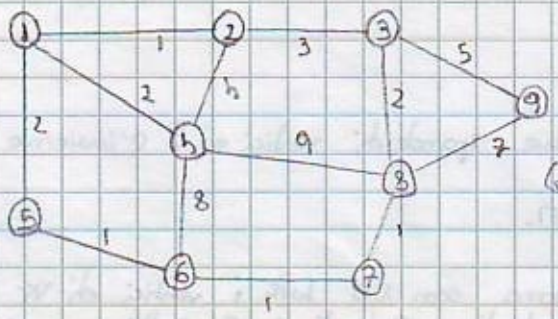
$S = \emptyset$

→ Attenzione: siamo tornato al passo 2.



Abbiamo trovato P, in quanto W_2 non è accoppiato.

Esiste però un'applicazione del problema dell'accoppiamento per un grafo, non bipartito. Questa applicazione prende il nome di problema DEL POSTINO CINESE. Sia G un grafo con lati di costo non negativo. Vogliamo trovare un ciclo euleriano di costo minimo. Si ricordi che un multigrafo è euleriano se e solo se è connesso e tutti i suoi vertici hanno grado pari. Consideriamo per esempio:



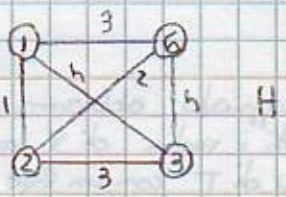
Si osservi che l'insieme di vertici di grado dispari in G ha cardinalità pari. Inoltre tutti i vertici di V' devono avere grado dispari nel grafo (V,S) mentre i rimanenti vertici devono avere grado pari in modo tale che G risulti Euleriano.

1) Trovare tutti i vertici di grado dispari.
2) Determinare i cammini minimi tra le coppie di vertici di grado dispari.
3) Costruire un grafo H con tali vertici, dove il peso del lato (v,w) è uguale alla lunghezza minima tra (v,w)
4) Trovare l'accoppiamento M perfetto a minimo di H
5) Per ogni lato di M, ricostruisci il cammino minimo duplicando gli archi in G, ottenendo così G'.
6) Trovare in G' un ciclo Euleriano.

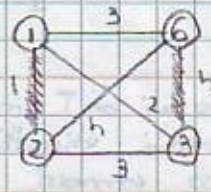
Per l'esempio si ha:

1) ①, ②, ③, ⑥ → Nodi di grado dispari.

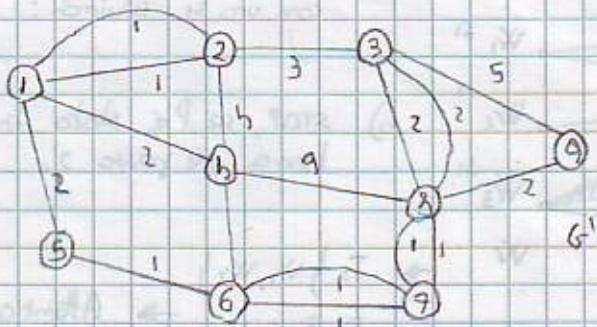
2)3)



4)



5)



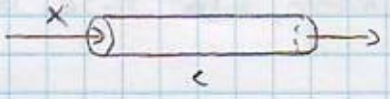
La complessità di un tale algoritmo è di siamo $\leq O(n^3)$. Si noti inoltre che il grafo G' ha tutti i nodi pari, quindi esiste un ciclo Euleriano V

Quando si lavora con i grafi spesso ci si trova di fronte ad un PROBLEMA DI FLUSSO. Supponiamo di avere:

Im breve supponiamo di avere un digrafo $G=(N,A)$ con n nodi ed m archi. I nodi speciali sono stati etichettati con s,t.

$\begin{cases} s = \text{NODO SORGENTE} \\ t = \text{NODO DESTINAZIONE} \end{cases}$

Ogni arco $a \in A$ possiede una capacità che viene indicata con $c(a)$ (simile al concetto di ugello di un arco). Si pensi per esempio ad una tubatura:



Sia x il FLUSSO entrante nel tubo. Il tubo ha capacità c.

Chiaramente il flusso che circolerà nel tubo sarà in quantità minore o uguale alla capacità del tubo stesso. Passiamo quindi, per analogia, dal tubo ad un arco. In un digrafo ci possono molti archi, e su ognuno di essi ci può essere un ben determinato flusso. Quindi sia:

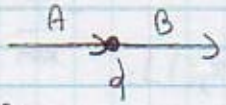
$X=[x_{ij}]$ la matrice dei flussi associati a ciascun arco (i,j) tale che:

$$0 \le x_{ij} \le c_{ij}$$

Questo significa che il flusso x_{ij} che circola nell'arco (i,j) può essere compreso tra il valore 0 (assenza di flusso) e la capacità dell'arco (flusso massimo). Quindi per definizione:

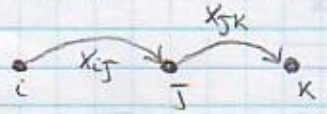
- un arco (i,j) si dice SATURO quando $\underline{x_{ij} = c_{ij}}$.
- un arco (i,j) si dice SCARICO quando $\underline{x_{ij} = 0}$.

Si consideri ora la seguente



q è un nodo, A e B due archi.

Il flusso che entra nel nodo q deve essere uguale al flusso che esce dal nodo q. Questa proprietà va sotto il nome di SOLENOIDALITÀ. Quindi:



$$\sum_{i=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{jk} = 0 \quad \text{se } j \neq s,t$$

$$\sum_{i=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{jk} = v \quad \text{se } j = t$$

$$\sum_{i=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{jk} = -v \quad \text{se } j = s$$

Si pensi per analogia alla legge di Kirchhoff delle correnti. Si noti che:

il flusso è solo uscente e vale $-v$ in quanto:

$$\sum_{i=1}^{m} \cancel{x_{is}}^{0} - \sum_{j=1}^{m} \cancel{x_{sj}}^{v} = -v$$

Analogamente:

Il flusso è pari a: $\sum_{i=1}^{n} x_{it} - \sum_{j=1}^{n} x_{tj} = v$

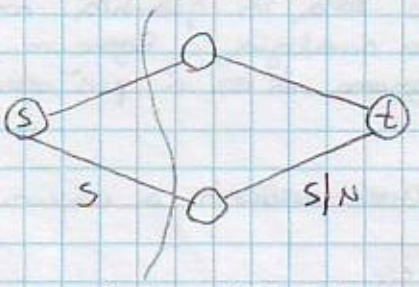
Si consideri ora un digrafo, e sia P un cammino non orientato da 's' a 't'.

Un arco (i,j) in P si dice in AVANTI se diretto da 's' a 't', mentre si dice a RITROSO se vale il viceversa.

Il cammino P è un CAMMINO AUMENTANTE rispetto ad un dato flusso $X=[x_{ij}]$ se $x_{ij} < c_{ij}$ per tutti gli archi in avanti di P, mentre per gli archi a ritroso, si ha: $x_{ij} > 0$. Un S-T TAGLIO è identificato da un insieme $S \subset N$ tale che:

$$s \in S \quad e \quad t \in N/S.$$

Graficamente:



$S \cup N/S = N$ NB: $T = N/S$.

La capacità di un taglio (S,T) è data da: $C(S,T) = \sum_{i \in S} \sum_{j \in T} c_{ij}$

Per ogni taglio (S,T) si ha:

$$v = \sum_{(i,j) \in F_S} x_{ij} - \sum_{(i,j) \in B_S} x_{ij}$$

Indichiamo infatti con F_S l'insieme degli archi in avanti e con B_S l'insieme degli archi a ritroso. Vediamone la dimostrazione:

$$v = \sum_{j \in T} \left(\sum_{i=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{jk} \right) \Rightarrow v = \sum_{j \in T} \left[\sum_{i \in S} x_{ij} + \sum_{i \in T} x_{ij} - \sum_{k \in S} x_{jk} - \sum_{k \in T} x_{jk} \right] =$$

$$= \sum_{j \in T} \left[\sum_{i \in S} (x_{ij} - x_{ji}) + \sum_{i \in T} (x_{ij} - x_{ji}) \right] =$$

$$= \sum_{j \in T} \sum_{i \in S} (x_{ij} - x_{ji}) = \sum_{(i,j) \in F_S} x_{ij} - \sum_{(i,j) \in B_S} x_{ji}$$

Per ogni taglio $(S-T)$ e flusso X, si ha: $C(S,T) \geq v$.

Si dimostra quest'ultimo lemma nel seguente modo: $v = \sum_{(i,j) \in F_S} x_{ij} - \sum_{(i,j) \in B_S} x_{ij} \leq \sum_{(i,j) \in F_S} c_{ij} - 0 = C(S,T)$

Il valore v del flusso X è massimo se e solo se non esistono da 's' a 't' cammini aumentanti rispetto a X.

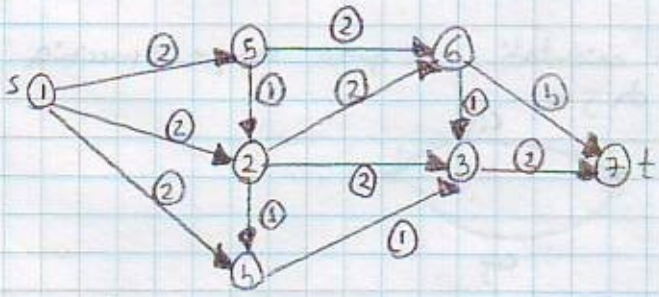
Inoltre se tutte le capacità c_{jk} sono intere, esiste un flusso X ammissibile e massimo di valore intero. Infine il valore massimo di un flusso da 's' a 't' è uguale alla capacità di un $(S-T)$ taglio di capacità minima. Analizziamo ora l'algoritmo del flusso massimo. Si ricordino le proprietà fondamentali viste:

$$\varphi(S) = \sum_{(i,j) \in F_S} x_{ij} - \sum_{(i,j) \in B_S} x_{ij} \Rightarrow \varphi(\{s\}) = \sum_{(s,j) \in F_S} x_{sj} - \sum_{(i,s) \in B_S} x_{is}$$

L'algoritmo MAX-FLOW funziona così:

1) Per ogni arco in avanti, controlla se $x_{ij} < c_{ij}$
2) Se avviene ciò, costruisco la rete incrementale e trovo cammino P da 's' a 't'.
3) Se P non esiste, fine.

Tale algoritmo è stato descritto in maniera alquanto semplicistica. Vediamolo con un esempio:

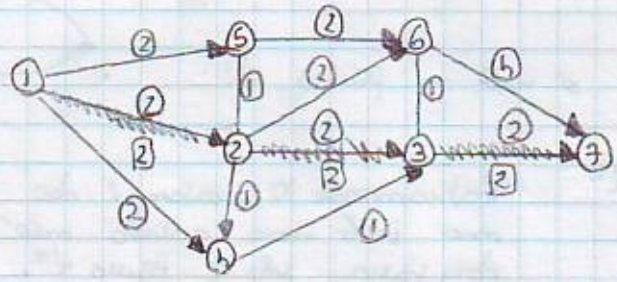


(a) = COSTO SULL'ARCO.

Innanzitutto una RETE INCREMENTALE o GRAFO AUSILIARIO è un grafo dove da ogni arco (i,j) di A possono originarsi al massimo due archi del grafo ausiliario:

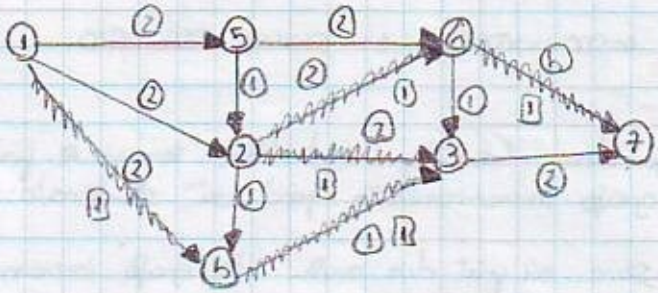
1) $a^+(i,j)$ se $x_{ij} < c_{ij}$ → $l_{ij} = c_{ij} - x_{ij}$ dove l_{ij} = lunghezza arco (i,j).
2) $a^-(j,i)$ se $x_{ij} > 0$ → $l_{ij} = x_{ij}$

Supponiamo che ie flusso, in tutti i nostri problemi, sia inizialmente nullo. Vediamo i cammini aumentanti:



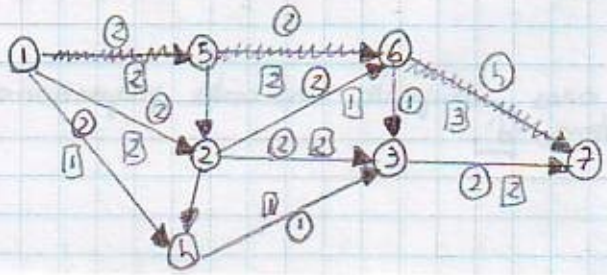
$\delta = 2$ → unità di flusso inviate

[a] = FLUSSO



$\delta = 1$

NB: L'arco (2,3) viene percorso a ritroso e quindi si ha che su tale arco si può costruire un cammino P.



$\delta = 2$