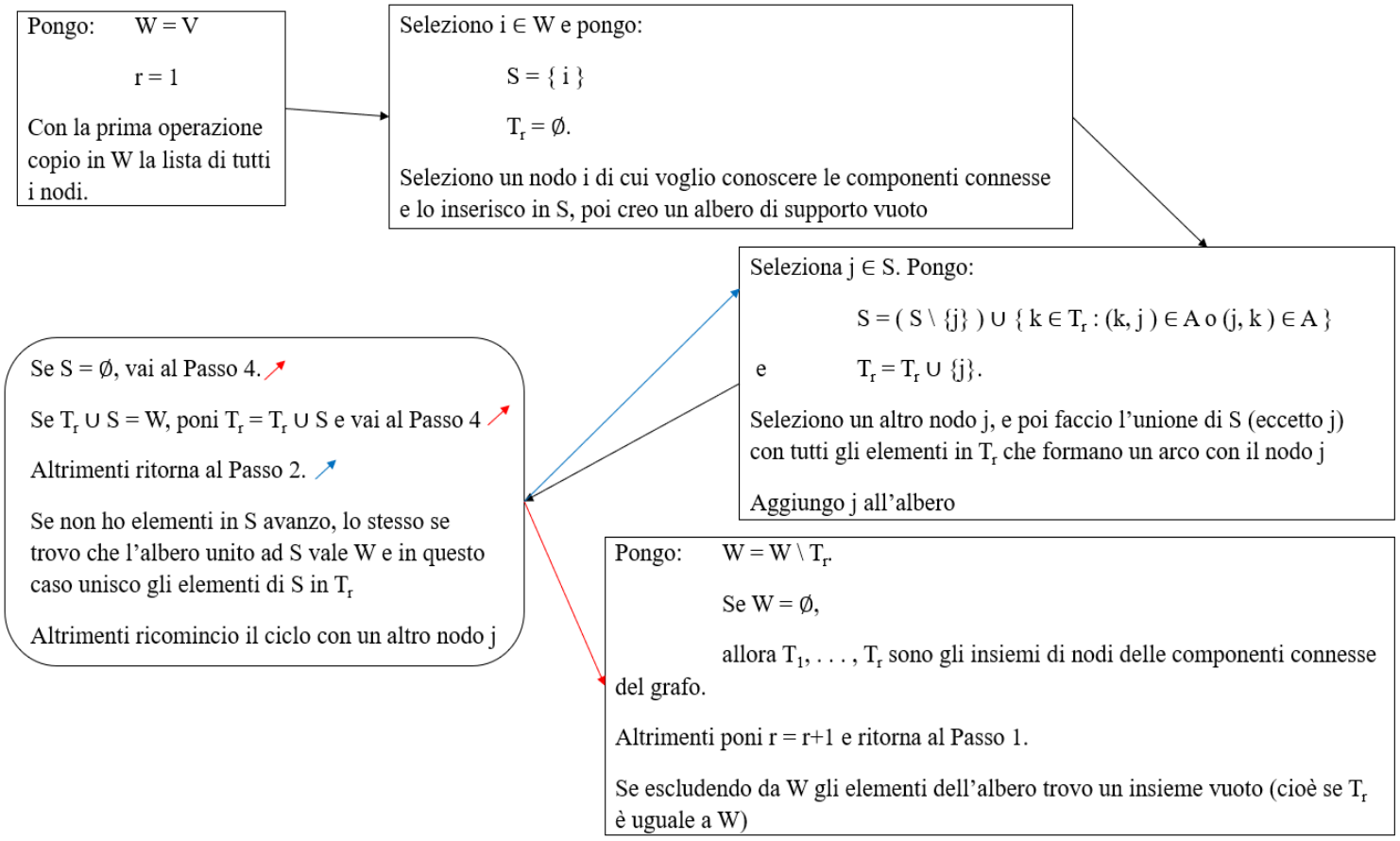
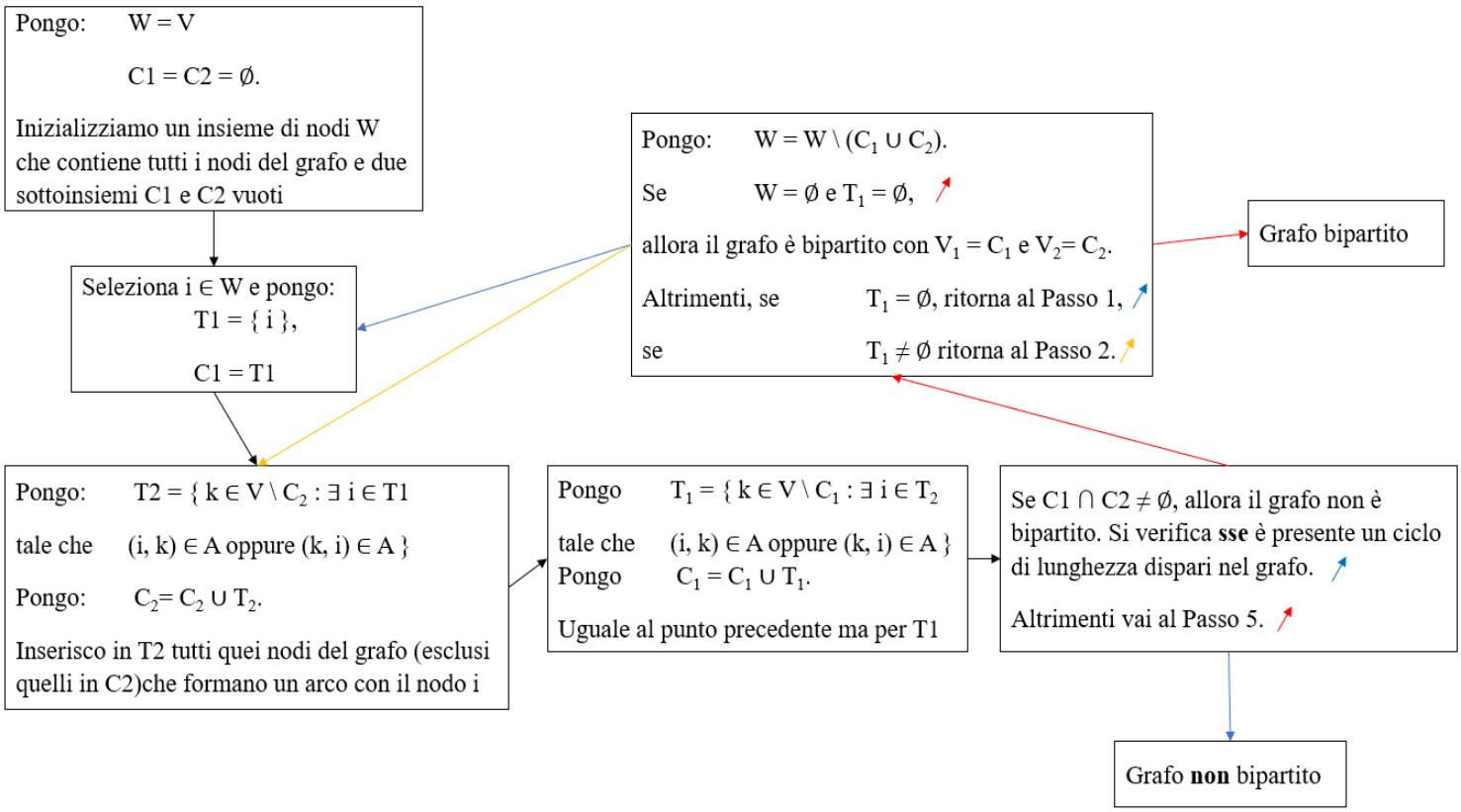
In questo file cerco di spiegare in un modo più semplici e capibile alcuni algoritmi trattati nel corso.

Consiglio di dare una letta prima alla teoria in modo da capire i vari termini utilizzati. Per capire bene gli algoritmi consiglio di tenere vicino un esempio pratico mentre li si studia

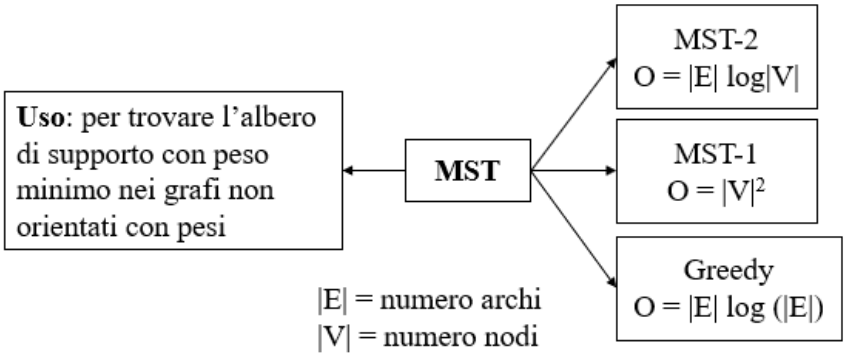
GRAFI

Individuazione componenti connesse:

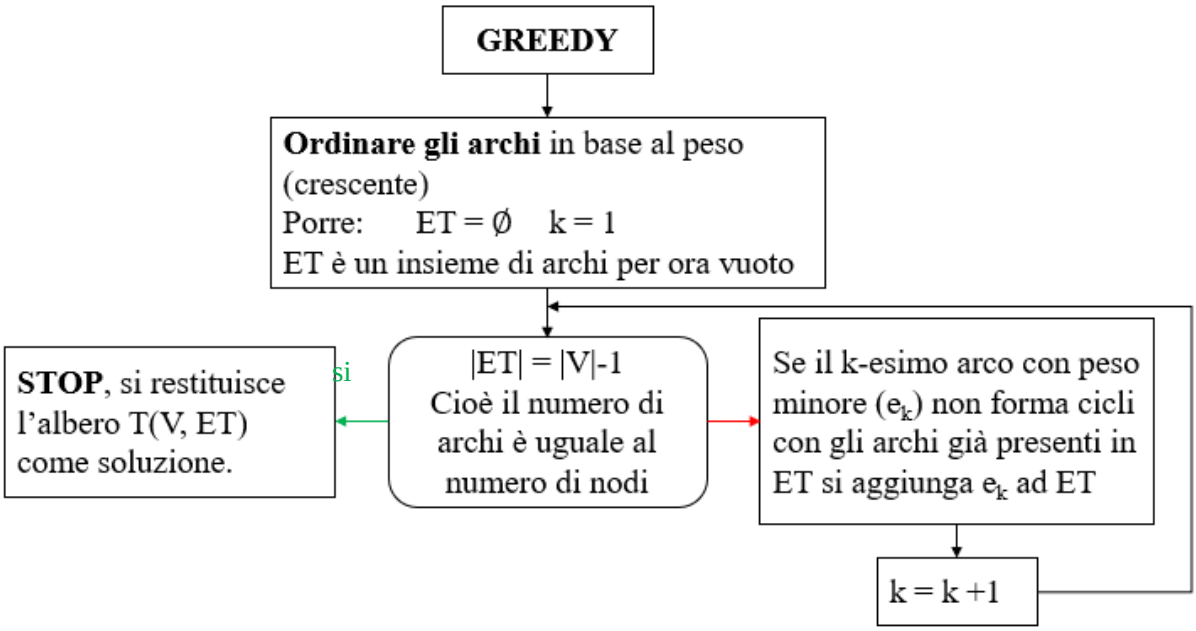




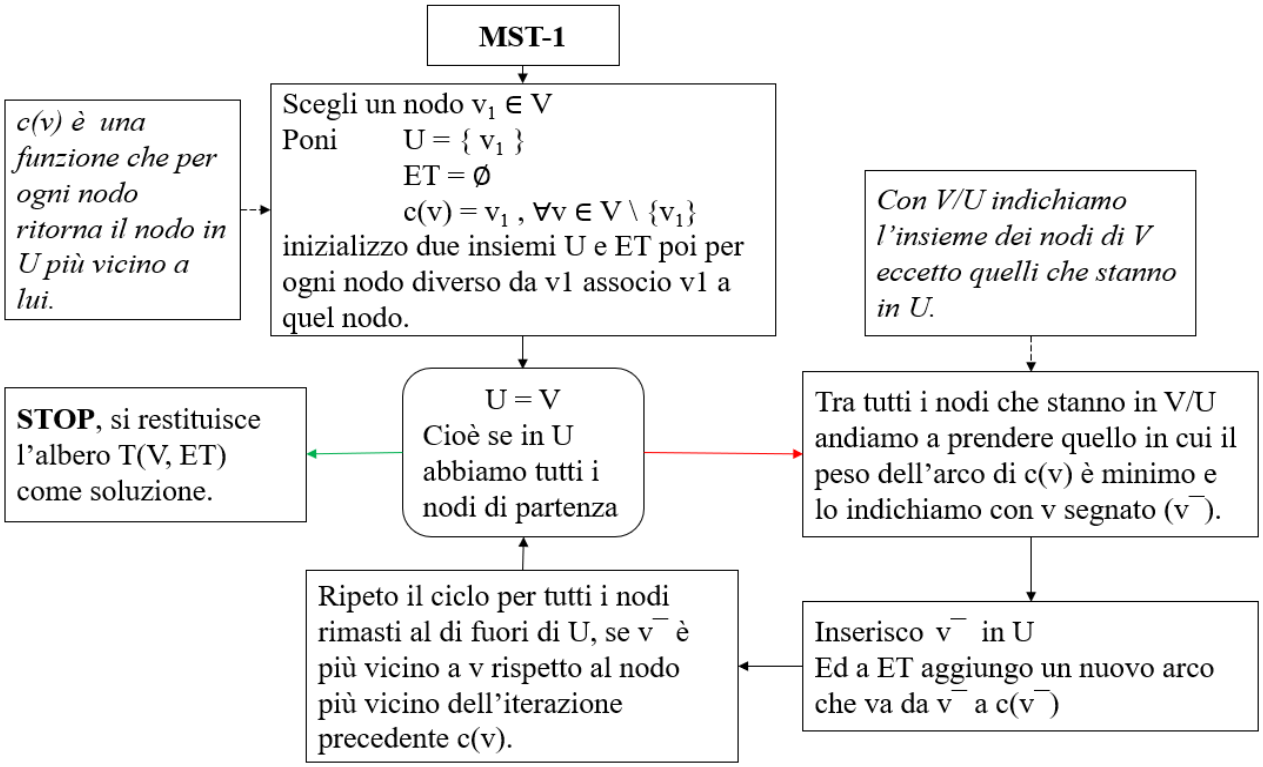
MST



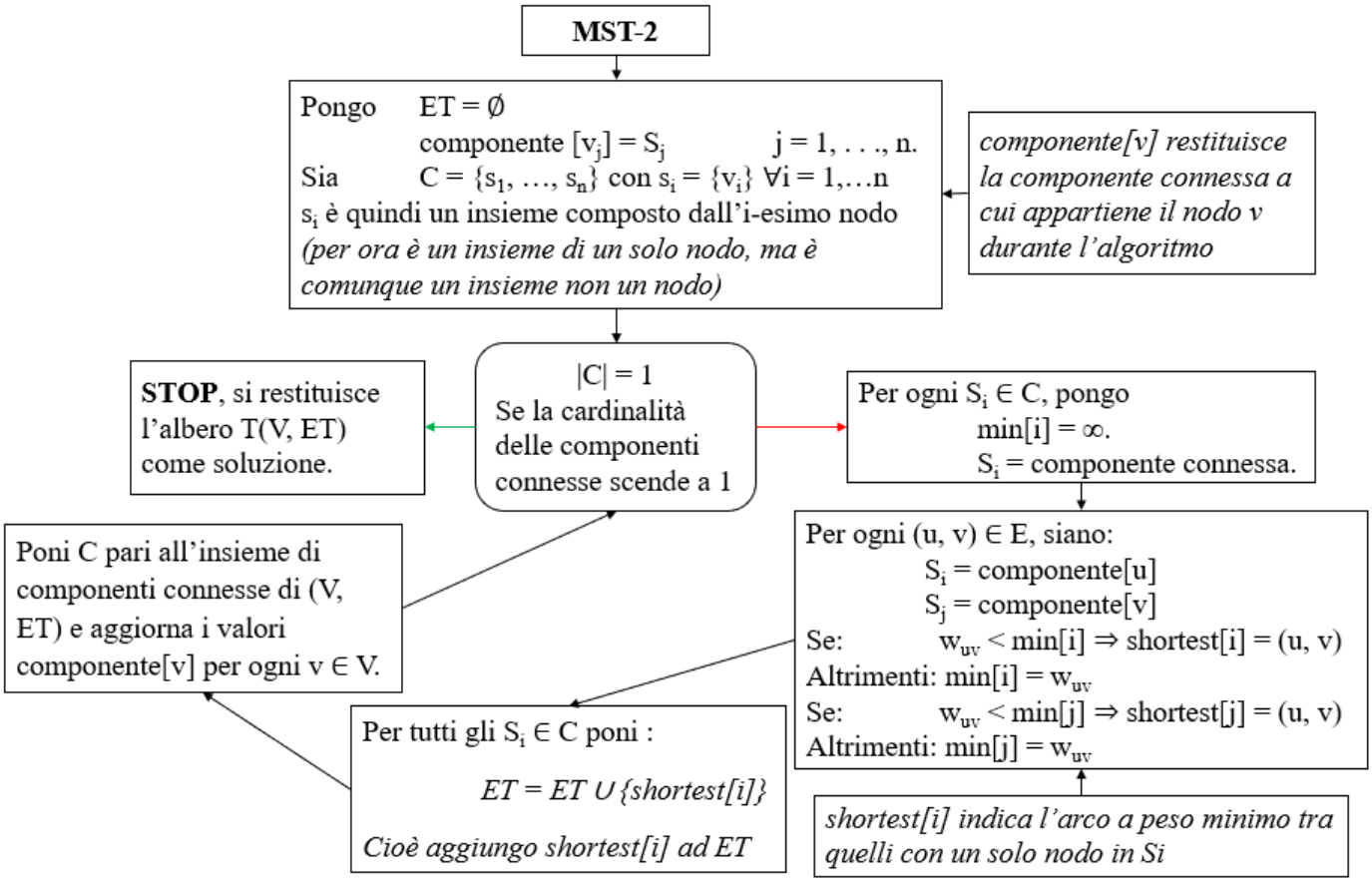
GREEDY



MST-1



MST-2



Spiegazione del procedimento centrale: quello che devo fare è trovare per ciascuna componente tutti quei archi che la "toccano". Nella tabella è mostrato un esempio, come si vede per ogni componente ho tutti quei archi che banalmente contengono quella componente. Dopo di che elimino quelli con peso maggiore e tengo solo quelli con il peso minore.

Componente	Shortest	Min
S1	(1,2)	3
S2	(1,2)	3
S3	~~(1,3)~~ (2,3)	~~5~~ 4
S4	~~(1,4)~~ ~~(2,4)~~ (3,4)	~~9~~ ~~8~~ 7
S5	~~(1,5)~~ (2,5)	~~20~~ 10

Metto tutti i rimanenti in ET; ET = (1,2);(2,3);(3,4);(2,5)

Dopo di che devo controllare la cardinalità delle componenti connesse e nel caso ripetere il ciclo.

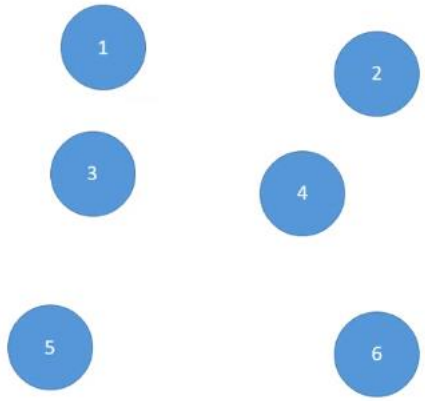
Siccome la cardinalità di ET è 1 termino.

Nell'esempio successivo è spiegato in modo leggermente diverso ma è la stessa cosa

esempio spiegato MST-2

$$\begin{bmatrix} \square & 2 & 8 & 8 & 10 & 3 \\ \square & \square & 12 & 7 & 11 & 1 \\ \square & \square & \square & 4 & 5 & 11 \\ \square & \square & \square & \square & 6 & 12 \\ \square & \square & \square & \square & \square & 10 \\ \square & \square & \square & \square & \square & \square \end{bmatrix}$$

Componente	Shortest	Min
$S_1 = \{1\}$	-	∞
$S_2 = \{2\}$	-	∞
$S_3 = \{3\}$	-	∞
$S_4 = \{4\}$	-	∞
$S_5 = \{5\}$	-	∞
$S_6 = \{6\}$	-	∞



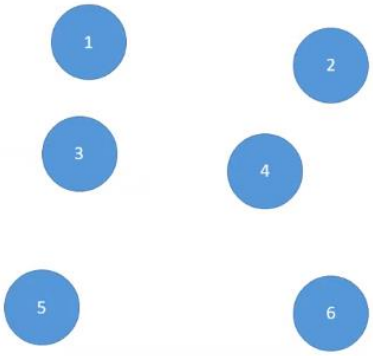
Quello che ci viene dato è la **tabella dei pesi** che è il nostro dato di partenza da qui dobbiamo partire a risolvere il nostro problema.

Per prima cosa **costruiamo la tabella** con tante righe quanti sono i nodi presenti , cioè il numero di righe nella nostra tabella dei pesi (quindi 6 per noi).

Poi **inseriamo in ogni riga la componente** relativa, che non è altro che un array da un solo elemento, quell'elemento è il nodo con lo stesso numero della riga (**riga 1 - nodo 1**). E mettiamo in min il valore infinito, questo è più un passaggio mentale ma che non ha troppa utilità è solo per ricordarci che li non c'è ancora un collegamento attivo.

$$\begin{bmatrix} \square & 2 & 8 & 8 & 10 & 3 \\ \square & \square & 12 & 7 & 11 & 1 \\ \square & \square & \square & 4 & 5 & 11 \\ \square & \square & \square & \square & 6 & 12 \\ \square & \square & \square & \square & \square & 10 \\ \square & \square & \square & \square & \square & \square \end{bmatrix}$$

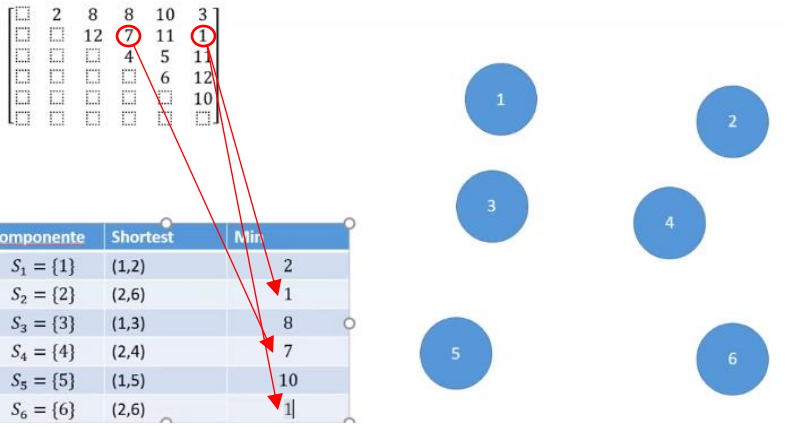
Componente	Shortest	Min
$S_1 = \{1\}$	(1,2)	2
$S_2 = \{2\}$	(1,2)	2
$S_3 = \{3\}$	(1,3)	8
$S_4 = \{4\}$	(1,4)	8
$S_5 = \{5\}$	(1,5)	10
$S_6 = \{6\}$	(1,6)	3



Il passo successivo è quello di inserire in shortest i collegamenti della prima riga della tabella dei pesi. Il primo ed il secondo sono per ora uguali.

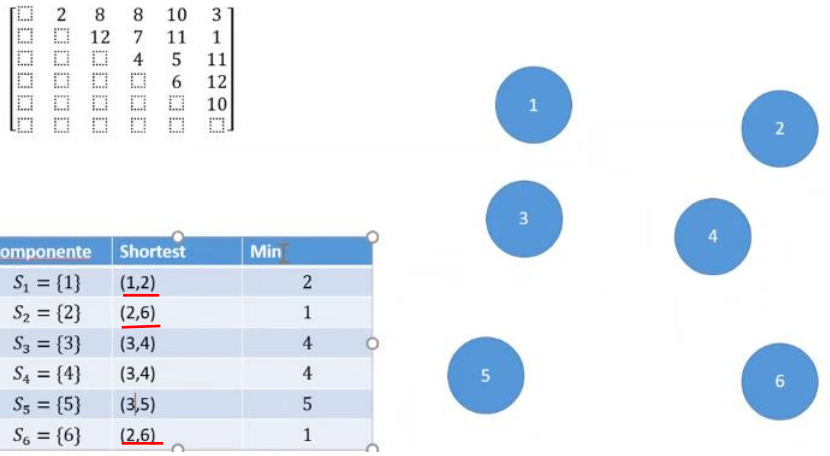
Quello che faccio ora è ripetere il ragionamento con i pesi nella seconda riga e cambiare i valori solo se il peso precedente è maggiore di quello attuale.

Nel nostro esempio notiamo che esiste un collegamento tra 2-6 di peso 1 che è minore del precedente collegamento 1-2 che aveva peso 2 e di quello tra 1-6 che aveva peso 3. Stesso ragionamento va fatto per il collegamento tra 1-4 (con peso 8) e quello tra 2-4 (peso 7).



Componente	Shortest	Min
$S_1 = \{1\}$	(1,2)	2
$S_2 = \{2\}$	(2,6)	1
$S_3 = \{3\}$	(1,3)	8
$S_4 = \{4\}$	(2,4)	7
$S_5 = \{5\}$	(1,5)	10
$S_6 = \{6\}$	(2,6)	1

Ripeto ora lo stesso ragionamento per la riga 3,4,5.



Componente	Shortest	Min
$S_1 = \{1\}$	(1,2)	2
$S_2 = \{2\}$	(2,6)	1
$S_3 = \{3\}$	(3,4)	4
$S_4 = \{4\}$	(3,4)	4
$S_5 = \{5\}$	(3,5)	5
$S_6 = \{6\}$	(2,6)	1

Ora dobbiamo **raggruppare le componenti connesse**. Cosa sono? In modo molto informale si può dire che sono tutti quei nodi che hanno degli archi comuni. Ad esempio una componente connessa è:

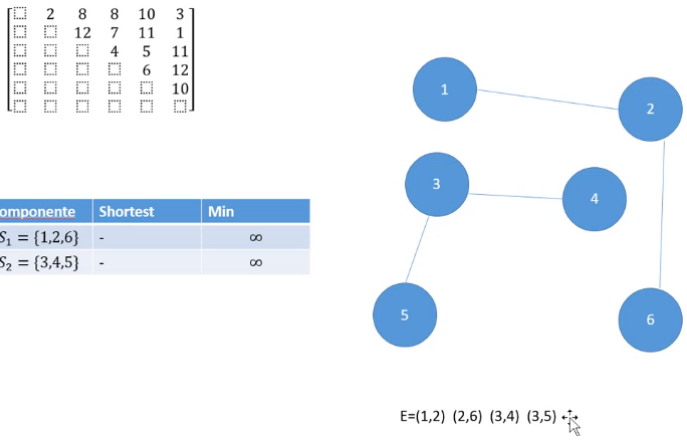
$S_1 = \{1,2,6\}$

Perché? Perché 1 e 2 sono legati dall'arco contenuto nella 1° riga, poi 2-6 sono legati tramite gli archi contenuti nella 2° e 6° riga. (le segno con il rosso)

Abbiamo però un'altra componente connessa che è formata da:

$S_2 = \{3,4,5\}$

Ora dobbiamo rispondere alla domanda, **abbiamo una sola componente connessa?** No, dunque dobbiamo ripetere il procedimento fatto finora ma con queste due nuove componenti.



Componente	Shortest	Min
$S_1 = \{1,2,6\}$	-	∞
$S_2 = \{3,4,5\}$	-	∞

E=(1,2) (2,6) (3,4) (3,5)

Ri-impostiamo di nuovo la tabella iniziale. E poi dobbiamo inserire i pesi, ma come fare?

Ora bisogna fare un po' di **attenzione** perché non tutti i pesi della prima riga possono essere utilizzati.

Iniziamo dicendo cosa ci indica la prima riga. Essa ci dice tutti i collegamenti da (e verso) il nodo 1.

La prima cella è vuota perché ci direbbe il peso del collegamento nodo 1 - nodo 1 che è nullo

La seconda indica il peso nodo 1 - nodo 2 che vale 8. Possiamo usare questo collegamento? **NO**, perché il nodo 2 fa parte della stessa componente del nodo 1 ma a noi interessa trovare un collegamento con la componente connessa S_2.

La terza cella indica il peso nodo 1 - nodo 3, possiamo usare questo collegamento? **SI**, visto che il nodo 3 fa parte della componente S_3. Il suo peso è 8

La quarta cella indica il peso nodo 1 - nodo 4, possiamo usare questo collegamento? **SI**, però ha lo stesso peso del precedente quindi non cambia nulla

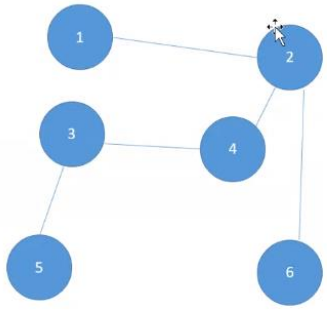
La quinta cella indica il peso nodo 1 - nodo 5, possiamo usare questo collegamento? **SI**, però ha un peso maggiore del precedente quindi tengo il peso precedente.

La sesta cella indica il peso nodo 1 - nodo 6, possiamo usare questo collegamento? **NO**, sono della stessa componente connessa.

Un ragionamento simile lo faccio per la riga 2, trovando che il collegamento 2-4 ha un peso di 7 che è inferiore.

	2	8	8	10	3
		12	7	11	1
			4	5	11
				6	12
					10

Componente	Shortest	Min
$S_1 = \{1,2,6\}$	(2,4)	7
$S_2 = \{3,4,5\}$	(2,4)	7



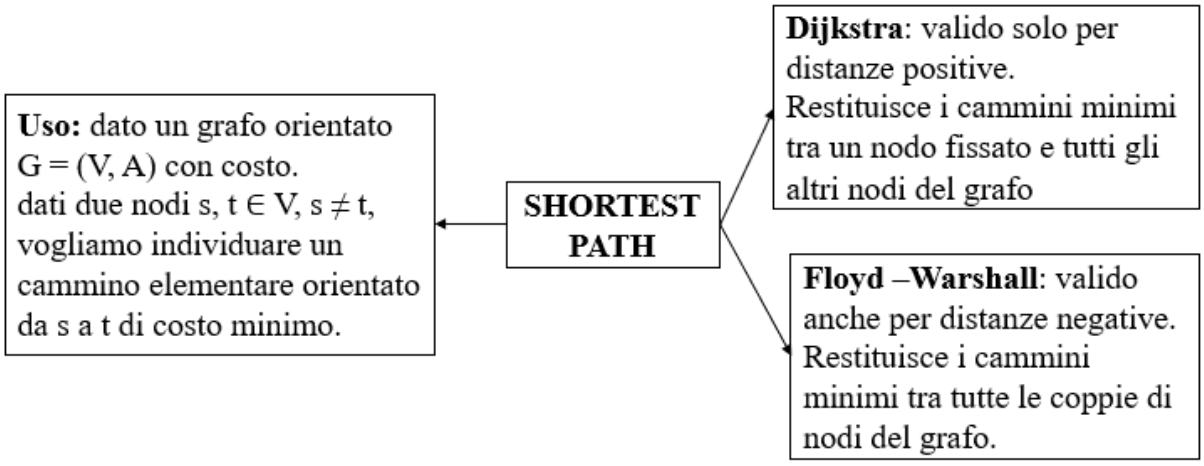
E=(1,2) (2,6) (3,4) (3,5) (2,4)

Siccome non ci sono altri collegamenti più leggeri significa che questo è il collegamento migliore.

Il risultato è una sola componente connessa S = {1,2,6,3,4,5} e l'albero di supporto risultante è dato dai collegamenti della prima e seconda fase insieme:

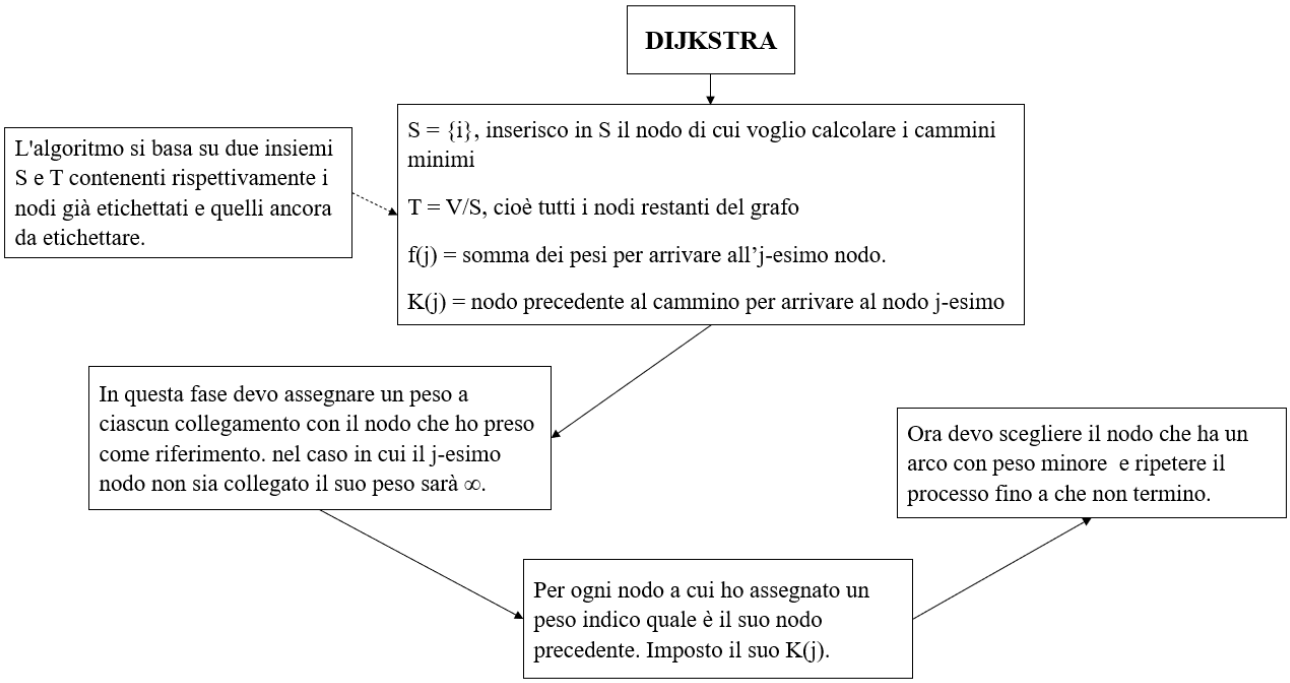
ET = (1,2) (2,6) (3,4) (3,5) (2,4)

SHORTEST PATH

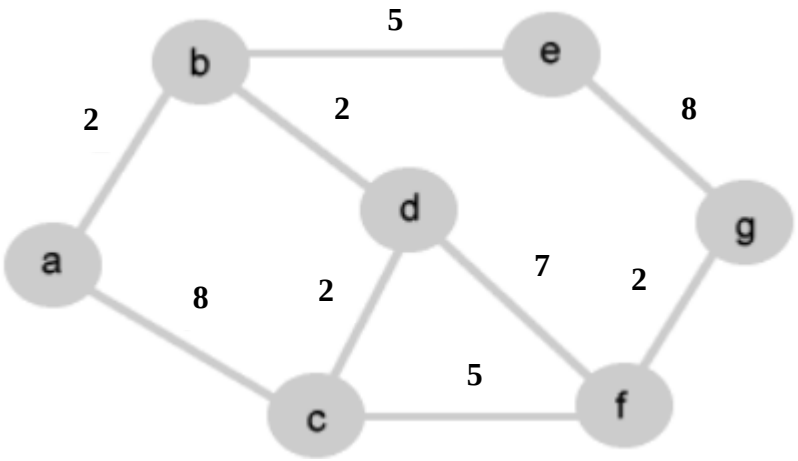


Dijkstra:

"*per questo algoritmo potrebbero essere diverse le lettere utilizzate e qualche termine perché per me spiegato nel modo in cui lo spiega il prof è molto complicato*. ***Inoltre il modo più facile per capire l'algoritmo è vedere l'esempio spiegato***"



Esempio spiegato:



Supponiamo di voler calcolare tutti i percorsi minimi del nodo a. quello che dobbiamo fare è:

S = {a}

T = {b, c, d, e, f, g}

f(x) = ∞, con x = b, c, d, e, f, g. (inizialmente il peso degli altri archi è ∞, è più una cosa teorica)

Inizio del **ciclo 1**:

1. per tutti i nodi che **non hanno un collegamento** con a, il loro peso rimane ∞.(quindi d, e, f, g)
2. per gli altri nodi che hanno un collegamento con a (cioè b, c), **solo se il peso è minore di quello precedente** vado a modificare il valore, in questo caso il peso precedente era ∞ quindi imposto un nuovo valore.

Imposto quindi:
   - f(b) = 2
   - f(c) = 8
3. ora devo indicare, **solo per i nodi connessi**, quale era il loro nodo precedente, che è a.
   K(b) = a
   K(c) = a
4. inserisco in S ora il nodo con peso minore tra quelli connessi ad a. in questo caso inserisco b in S visto che f(b) < f (c).

ripeto ora il ciclo che il nuovo nodo inserito in S, cioè con b.

Inizio del **ciclo 2**:

1. per tutti i nodi che **non hanno un collegamento** con b, il loro peso rimane il precedente (c, f, g)
2. per gli altri nodi che hanno un collegamento con b (cioè e, d), **solo se il peso è minore di quello precedente** vado a modificare il valore, in questo caso il peso precedente era ∞ quindi imposto un nuovo valore.
   Imposto quindi:
   - f(d) = f(b) + 2 = 4
   - f(e) = f(b) + 5 = 7
3. ora devo indicare, **solo per i nodi connessi**, quale era il loro nodo precedente, che è b.
   K(d) = b
   K(e) = b
4. inserisco in S ora il nodo con peso minore tra quelli connessi ad b. in questo caso inserisco d in S visto che f(d) < f (e).

ripeto ora il ciclo che il nuovo nodo inserito in S, cioè con d.

Inizio del **ciclo 3**:

1. per tutti i nodi che **non hanno un collegamento** con d, il loro peso rimane il precedente (g)
2. per gli altri nodi che hanno un collegamento con d (cioè c, f), **solo se il peso è minore di quello precedente** vado a modificare il valore.
   NB. Per c avevamo già un peso che avevamo calcolato nel primo ciclo, dunque ora sostituiamo il nuovo valore di f(c) solo se è minore del precedente (che era 8)
   Imposto quindi:
   - f(c) = f(d) + 2 = 6 (minore del precedente)
   - f(f) = f(d) + 7 = 11
3. ora devo indicare, **solo per i nodi connessi**, quale era il loro nodo precedente, che è b.
   K(c) = d
   K(f) = d
4. inserisco in S ora il nodo con peso minore tra quelli connessi ad d. in questo caso inserisco c in S visto che f(c) < f (f).

Dobbiamo continuare cosi, prendiamoci una pausa per valutare quello che abbiamo calcolato finora.

S = {a, b, d, c}

T = {e, f, g}

f(b) = 2 f(d) = 4 f(c) = 6

K(b) = a K(d) = b K(c) = d

ripeto ora il ciclo che il nuovo nodo inserito in S, cioè con c.

Inizio del **ciclo 4**:

1. per tutti i nodi che **non hanno un collegamento** con c, il loro peso rimane il precedente (g)
2. per gli altri nodi che hanno un collegamento con c (f), **solo se il peso è minore di quello precedente** vado a modificare il valore.
   NB. Per f avevamo già un peso che avevamo calcolato nel precedente ciclo, dunque ora sostituiamo il nuovo valore di f(f) solo se è minore del precedente (che era 11)
   Imposto quindi:
   - f(f) = f(c) + 5 = 11 (**NON è minore** del precedente)
3. siccome non ho nuovi pesi non devo neanche gestire il nodo precedente.
4. In questo caso non ci sono nuovi nodi da aggiungere

Il nodo che prendo ora in considerazione per proseguire il processo sarà quello che ha un peso minore contenuto in T, cioè e. (f(e) = 7, f(f) = 11, f(g) = ∞)

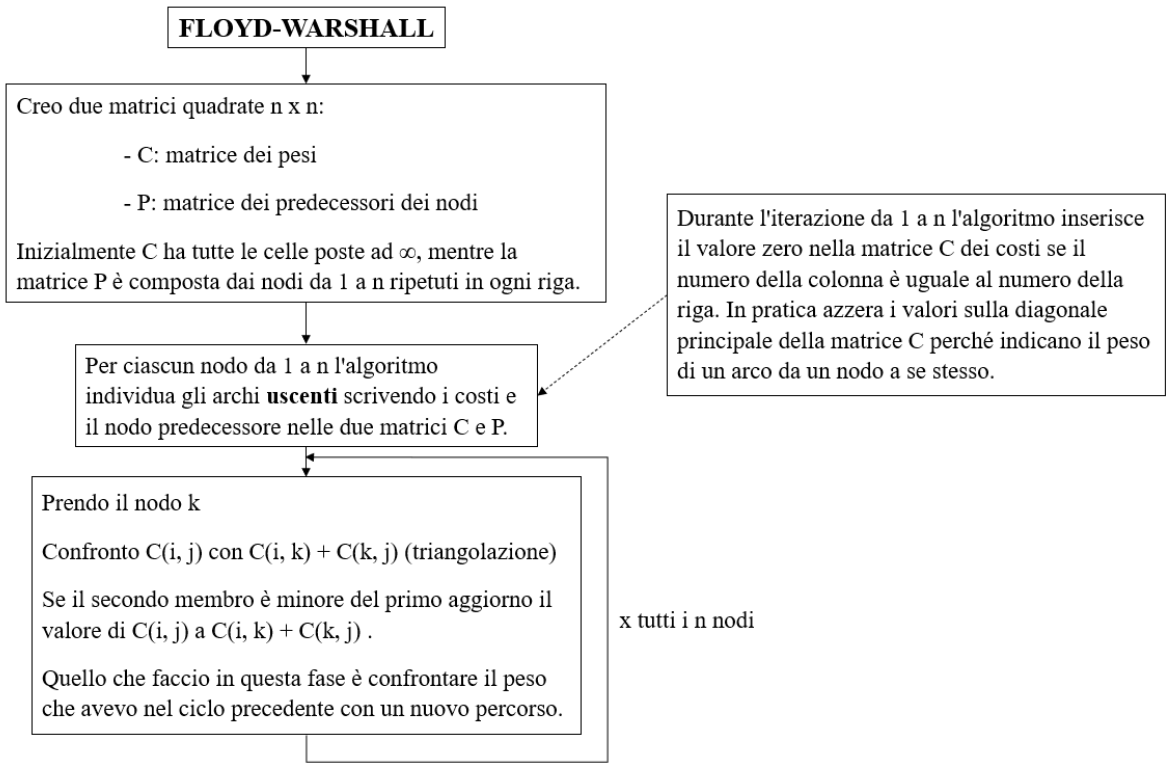
Ripeto di nuovo il ciclo per e ed poi per f. il risultato finale sarà:

Nodo	**Nodo precedente K(x)**	**Peso f(x)**
b	a	2
c	d	6
d	b	4
e	b	7
f	d	11
g	e	15

Durante il calcolo può essere comodo aggiornare di volta in volta questa tabella invece che tenere a mente i valori.

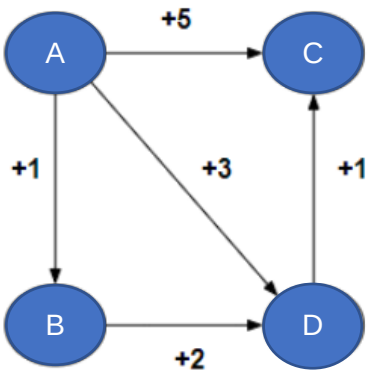
FLOYD-WARSHALL

Il risultato di questo algoritmo è una matrice n x n dove la cella (i, j) indica il peso del collegamento tra il nodo i e il j ed una matrice che indica i nodi predecessori.



Anche in questo caso come prima il modo più semplice per spiegare l'algoritmo è con un esempio.

Esempio spiegato:



supponiamo di avere il seguente grafo.

La prima cosa da fare è l'inizializzazione, dunque creare le due tabelle C e P.

Inizializzazione:

Matrice C

DA / A	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

Matrice P

DA / A	A	B	C	D
A	A	B	C	D
B	A	B	C	D
C	A	B	C	D
D	A	B	C	D

Individuazione degli archi uscenti:

in questa fase per ciascun nodo vado a **modificare il peso** solo degli archi uscenti e connessi con altri nodi. Ad esempio quello tra (A, B) con peso +1 oppure (A, C) con peso +5, ecc…

in questa stessa fase **azzero la diagonale**.

Matrice C

DA / A	A	B	C	D
A	0	1	5	3
B	∞	0	∞	2
C	∞	∞	0	∞
D	∞	∞	1	0

Inoltre devo aggiornare anche P con i nodi predecessori *("scrivo il numero della riga se ho modifica la cella nella tabella dei pesi"*)

DA / A	A	B	C	D
A	A	A	A	A
B	A	B	C	B
C	A	B	C	D
D	A	B	D	D

Fase di elaborazione:

in cosa consiste la triangolazione? La triangolazione è chiedersi se esiste un percorso migliore per arrivare da un nodo all'altro passando per uno (o più) nodi intermedi.

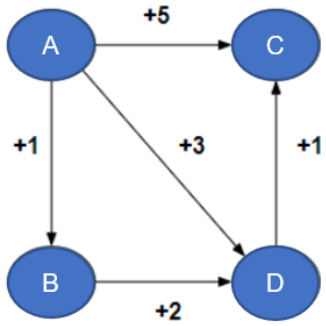
ES. nel nostro caso specifico si vedrà sviluppando i calcoli (ma lo si può vedere anche ad occhio nudo) come il percorso più "veloce" per passare da A ad C sia quello di passare dai nodi B e C (o da D) rispetto al percorso diretto.

nodo A quindi → k = A:

quello che dobbiamo fare in questa caso è valutare i pesi dei vari collegamenti ed modificarli se troviamo percorsi migliori. Abbiamo una tabella con n^2 righe (qui $4^2 = 16$), dove:

- nella **prima colonna** è indicato il **collegamento da valutare**
- nella **seconda colonna** è indicato il **costo dell'arco** (dato dal ciclo precedente che nel primo caso è quello dato dall'inizializzazione). Per riempire questa colonna basta copiare il valore che è nella cella (i, j) della tabella C
- nella **terza colonna** è indicato il **peso della triangolazione** con il nodo k.

(i, j)	C(i, j)	C(i, k) + C(k, j)
(A, A)	0	C(A, A) + C(A, A) = 0 + 0 = 0
(A, B)	1	C(A, A) + C(A, B) = 0 + 1 = 1
(A, C)	5	C(A, A) + C(A, C) = 0 + 5 = 5
(A, D)	3	C(A, A) + C(A, D) = 0 + 3 = 3
(B, A)	∞	C(B, A) + C(A, A) = ∞ + 0 = ∞
(B, B)	0	C(B, A) + C(A, B) = ∞ + 1 = ∞
(B, C)	∞	C(B, A) + C(A, C) = ∞ + 5 = ∞
(B, D)	2	C(B, A) + C(A, D) = ∞ + 3 = ∞
(C, A)	∞	C(C, A) + C(A, A) = ∞ + 0 = ∞
(C, B)	∞	C(C, A) + C(A, B) = ∞ + 1 = ∞
(C, C)	0	C(C, A) + C(A, C) = ∞ + 5 = ∞
(C, D)	∞	C(C, A) + C(A, D) = ∞ + 3 = ∞
(D, A)	∞	C(D, A) + C(A, A) = ∞ + 0 = ∞
(D, B)	∞	C(D, A) + C(A, B) = ∞ + 1 = ∞
(D, C)	1	C(D, A) + C(A, C) = ∞ + 5 = ∞
(D, D)	0	C(D, A) + C(A, D) = ∞ + 3 = ∞



DA / A	A	B	C	D
A	0	1	5	3
B	∞	0	∞	2
C	∞	∞	0	∞
D	∞	∞	1	0

In questo caso (ma di solito è sempre così per il primo nodo) non ci sono cambiamenti.

Sviluppando la tabella si nota che anche per k = B, e k = C, non ci sono cambiamenti quindi per brevità salto direttamente al caso di k = D.

(i, j)	C(i, j)	C(i, k) + C(k, j)
(A, A)	0	C(A, D) + C(D, A) = 3 + 0 = 3
(A, B)	1	C(A, D) + C(D, B) = 3 + ∞ = ∞
(A, C)	5	C(A, D) + C(D, C) = 3 + 1 = 4
(A, D)	3	C(A, D) + C(D, D) = 3 + 0 = 3
(B, A)	∞	C(B, D) + C(D, A) = 2 + ∞= ∞
(B, B)	0	C(B, D) + C(D, B) = 2 + ∞ = ∞
(B, C)	∞	C(B, D) + C(D, C) = 2 + 1 = 3
(B, D)	2	C(B, D) + C(D, D) = 2 + 0 = 2
(C, A)	∞	C(C, D) + C(D, A) = ∞ + ∞ = ∞
(C, B)	∞	C(C, D) + C(D, B) = ∞ + ∞ = ∞

(C, C)	0	C(C, D) + C(D, C) = ∞ + 1 = ∞
(C, D)	∞	C(C, D) + C(D, D) = ∞ + 0 = ∞
(D, A)	∞	C(D, D) + C(D, A) = 0 + ∞ = ∞
(D, B)	∞	C(D, D) + C(D, B) = 0 + ∞ = ∞
(D, C)	1	C(D, D) + C(D, C) = 0 + 1 = 1
(D, D)	0	C(D, D) + C(D, D) = 0 + 0 = 0

Abbiamo trovato dunque due archi in cui la triangolazione e minore del percorso diretto, dobbiamo quindi modificare le tabelle:

C:

DA / A	**A**	**B**	**C**	**D**
A	0	1	4	3
B	∞	0	3	2
C	∞	∞	0	∞
D	∞	∞	1	0

P:

DA / A	**A**	**B**	**C**	**D**
A	A	A	D	A
B	A	B	D	B
C	A	B	C	D
D	A	B	D	D

Va inserito nelle caselle il valore di k in cui è variato il percorso.