* Si discuta la complessità computazionale dell’algoritmo di Bellman-Ford.
  + L’algoritmo di Bellman-Ford ha una complessità ), in quanto viene fato un numero di iterazioni pari al numero di nodi con un ciclo che esamina le etichette duali su tutti gli archi . Qualora esistesse un ciclo negativo (abbiamo aggiornato almeno un’etichetta) oppure tutte le etichette sono stabili, l’algoritmo termina la sua esecuzione.
* Si discuta la complessità computazionale dell’algoritmo di Dijkstra per il problema del cammino minimo
  + Il prof riporta: “Si consiglia di riportare i passi dell’algoritmo di Dijkstra e, quindi, discuterne la complessità come fatto nelle dimostrazione della Proprietà 10 alla fine del paragrafo 6.1 delle dispense sugli algoritmi per problemi di cammino minimo. Quindi si può approfondire con qualche dettaglio sull’influenza della scelta di opportune strutture dati per migliorare l’efficienza, come dal paragrafo 6.2 delle stesse dispense.”

Per la complessità:

* consideriamo una complessità lineare di analisi di tutti i nodi
* cerchiamo il minimo e dobbiamo trovare un minimo dipendente dalla cardinalità di tutti gli elementi; *usando una lista*, avremo
* controlliamo gli archi che escono da un solo nodo, con
* la complessità sarebbe
* calcolando invece la complessità ammortizzata (cioè, considerando tutte le iterazioni e nel caso peggiore; se le operazioni costose sono poco frequenti, compenso “ammortizzando” con operazioni poco costose in modo deterministico), consideriamo ogni volta gli archi che escono dall’etichetta ottima; controlliamo *al massimo una volta* tutti gli archi.

Avremo quindi

La complessità finale, usando e come liste è

Implementando invece e come heap, possiamo fare la ricerca di minimo in tempo costante, che ora ci costa almeno , dato che se fosse min-heap, sta nella radice.

* Per creare un heap (*create*), costerà
* Per l’inserimento (*insert*), dovremo mantenere la proprietà di heap (facendo in modo che il nodo padre sia più grande/più piccolo di tutti i nodi figli e sia bilanciato, quindi con numero minimo di figli), con costo
* Una volta trovato il minimo (*find-min*), dobbiamo anche eliminare il nodo radice e ristabilire tutto lo heap.
* Quindi, eliminare il minimo (*delete-min*) costerà
* L’aggiornamento dell’etichetta (*decrease-key*) costerà

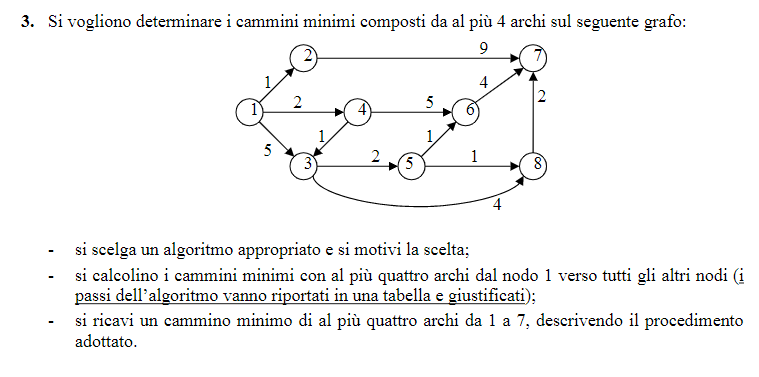
La complessità finale, usando e come heap è

Quindi, avremo ; tuttavia:

* se avessimo a che fare con un grafo completo/denso, meglio implementare con le liste;
* se avessimo a che fare con un grafo sparso, meglio implementare con gli heap.

Se usassimo un Fibonacci heap, si dimostra che la complessità sarà .

Simulazione Esame 2018/2019 – Esercizio 3 Grafi Raccolta “Esercizi vari”



1. Nella scelta dell’algoritmo, notiamo che ci viene dato un massimo numero di archi da dover rispettare, pertanto si può applicare solo l’algoritmo di Bellman – Ford, l’unico che dà la possibilità di calcolo dei cammini minimi sulla base di un massimo numero di archi. Applicheremo Bellman – Ford fermandoci alla quarta iterazioni, con un numero di archi e iterazioni pari a
2. Ora, il calcolo dei cammini minimi, riportati in tabella (controllando ad ogni passo miglioramenti rispetto alla riga precedente e segnando in colonna apposita gli aggiornamenti effettuati; i predecessori sono segnati come semplici pedici senza parentesi):

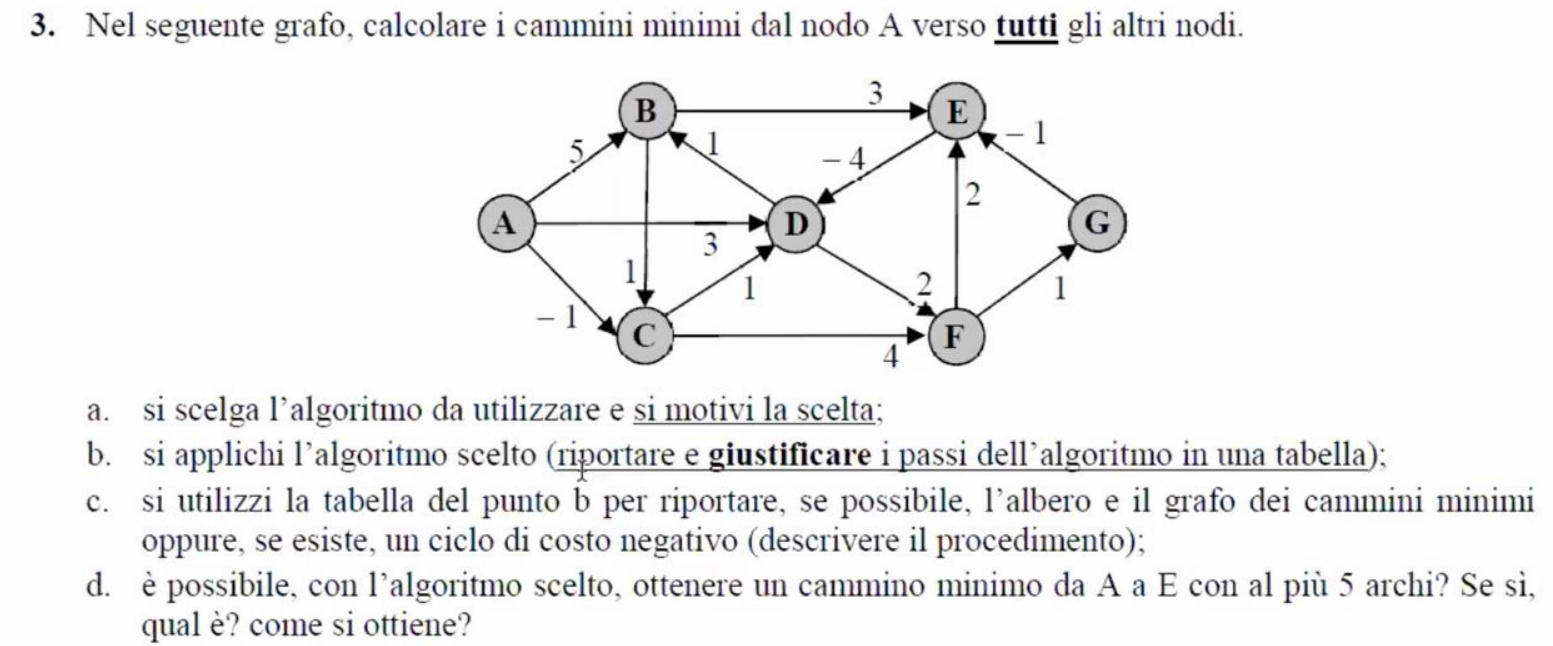
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo 1 | Nodo 2 | Nodo 3 | Nodo 4 | Nodo 5 | Nodo 6 | Nodo 7 | Nodo 8 | Aggiornamenti |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

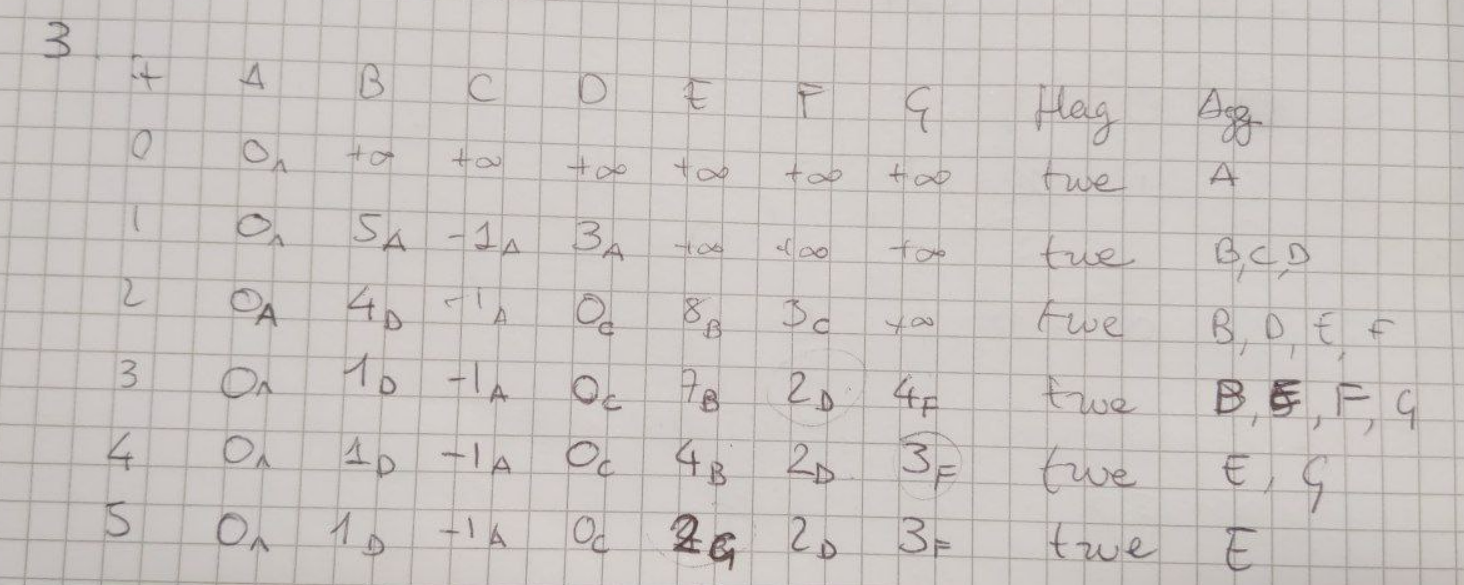
Le etichette di una riga sono ottenute controllando i vincoli duali su tutti gli archi uscenti dai nodi “aggiornati” della riga (iterazione) precedente secondo la dove è uno degli archi uscenti da un nodo aggiornato all’iterazione precedente, l’etichetta corrente (sulla riga corrente) del nodo , è l’etichetta del nodo all’iterazione (riga) precedente e è il costo dell’arco .

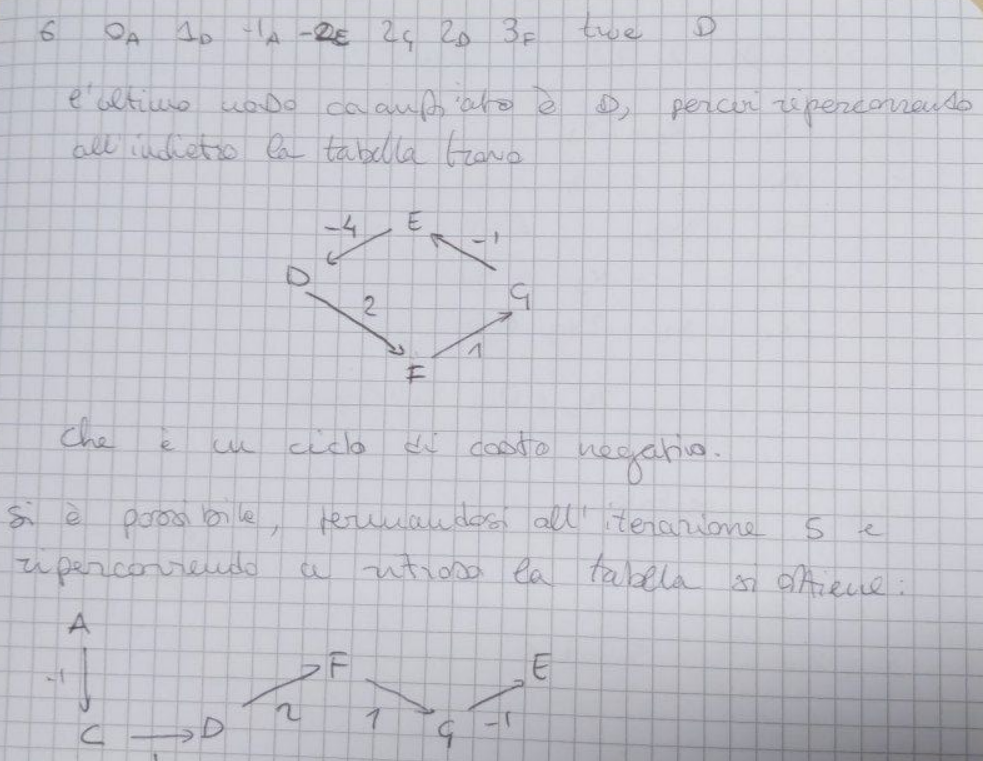
Grazie al fatto di utilizzare l’etichetta del nodo precedente, viene assicurata la scelta del cammino minimo con archi (qua si può fare un qualsiasi esempio numerico dove si va a scegliere, prendendo ad esempio l’ultima iterazione, qui un cammino che ha costo migliore considerando l’etichetta precedente e non quella corrente, qui , dimostrando la validità di quanto fatto).

1. Un cammino minimo con al massimo archi da viene fatta seguendo la catena dei predecessori, quindi, partendo dal nodo (ultimo nodo in generale) con si considera quindi:

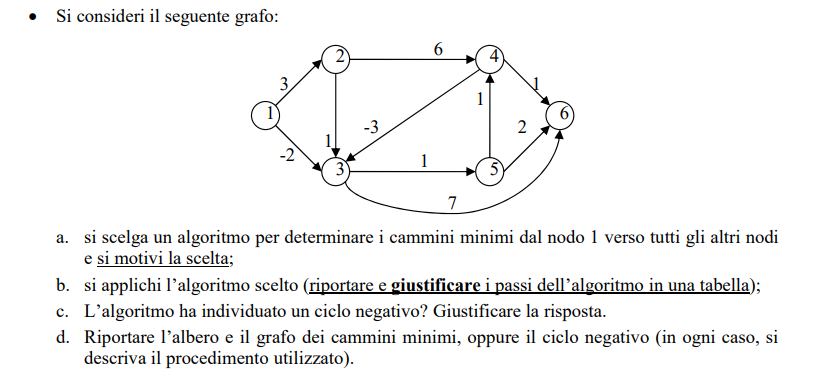
Questo equivale al percorso , con costo 9.

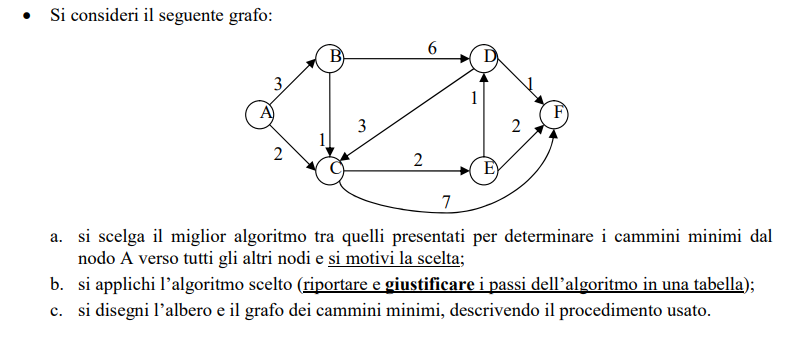


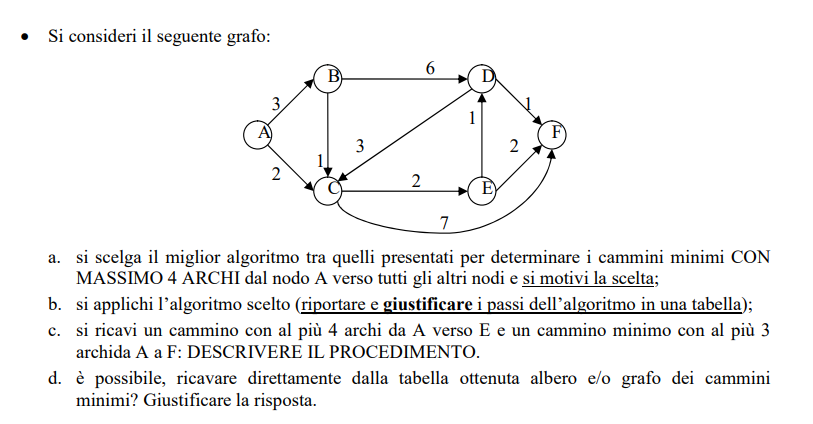




Carrellata di Esercizi da Raccolta “Esercizi vari”







1. Nella scelta dell’algoritmo, notiamo che ci viene dato un massimo numero di archi da dover rispettare, pertanto si può applicare solo l’algoritmo di Bellman – Ford, l’unico che dà la possibilità di calcolo dei cammini minimi sulla base di un massimo numero di archi. Applicheremo Bellman – Ford fermandoci alla quarta iterazioni, con un numero di archi e iterazioni pari a
2. Ora, il calcolo dei cammini minimi, riportati in tabella (controllando ad ogni passo miglioramenti rispetto alla riga precedente e segnando in colonna apposita gli aggiornamenti effettuati; i predecessori sono segnati come semplici pedici senza parentesi):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo A | Nodo B | Nodo C | Nodo D | Nodo E | Nodo F | Aggiornamenti |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

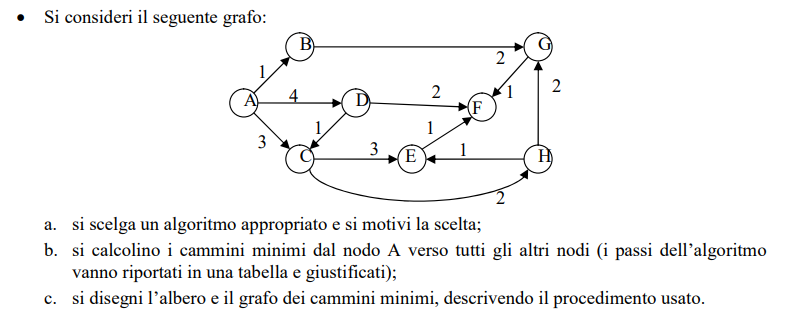
Le etichette di una riga sono ottenute controllando i vincoli duali su tutti gli archi uscenti dai nodi “aggiornati” della riga (iterazione) precedente secondo la dove è uno degli archi uscenti da un nodo aggiornato all’iterazione precedente, l’etichetta corrente (sulla riga corrente) del nodo , è l’etichetta del nodo all’iterazione (riga) precedente e è il costo dell’arco .

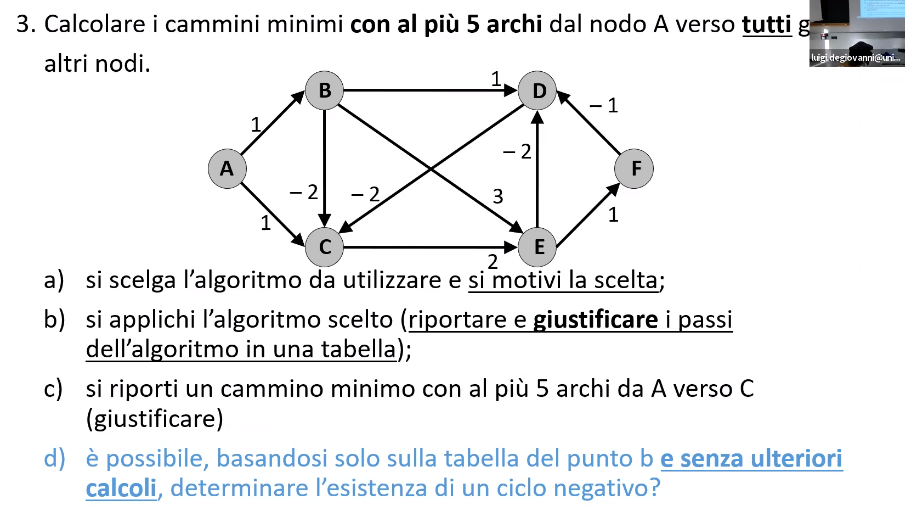
1. Un cammino minimo con al più archi da verso , seguendo la catena dei predecessori e considerano lo stesso , viene dato da con costo

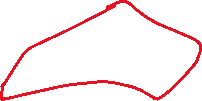
Un cammino minimo con al più archi da verso , considerando lo stesso e seguendo la catena dei predecessori, viene dato da con costo . Usando solo i predecessori, non abbiamo un albero dei cammini minimi.

Per l’individuazione di entrambi i cammini minimi, seguo come detto i predecessori partendo dal nodo indicato e diminuendo di 1, considerando la diminuzione delle iterazioni ad ogni passo e guardando la riga precedente. Infatti, nel caso si risolva un problema di cammino minimo con un massimo numero di hop, non si può parlare di albero dei cammini minimi.

1. L’albero dei cammini mini può essere ottenuto dalla tabella seguendo la catena dei predecessori (solo nel caso della prima richiesta), mentre per quanto riguarda il grafo, questo viene dato da tutti e soli i cammini minimi; in questo caso specifico, coincide con il percorso con al più 3 archi, in quanto quello da 4 non può essere considerato dati i motivi enunciati sopra.







a)

Nella scelta dell’algoritmo, notiamo che ci viene dato un massimo numero di archi da dover rispettare, pertanto si può applicare solo l’algoritmo di Bellman – Ford, l’unico che dà la possibilità di calcolo dei cammini minimi sulla base di un massimo numero di archi. Applicheremo Bellman – Ford fermandoci alla quarta iterazioni, con un numero di archi e iterazioni pari a

b)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo A | Nodo B | Nodo C | Nodo D | Nodo E | Nodo F | Aggiornamenti |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

c) Un cammino minimo (con al più 5 archi) può essere individuato seguendo la catena dei predecessori

che equivale al percorso con costo

A livello di algoritmo, si termina con *flag\_aggiornato=true*, pertanto esiste un ciclo negativo, in quanto ad ogni iterazione è stata aggiornata almeno un’etichetta.

d) Basandosi solo sulla tabella e senza ulteriori calcoli, è possibile determinare il ciclo negativo perché siamo arrivati “fino in fondo” con i calcoli.



e) È possibile disegnare albero e grafo dei cammini minimi? No, in quanto abbiamo etichette instabili (no soluzione ammissibile duale e quindi no cammini minimi).

Dimostriamo ad esempio esiste il ciclo ; come tale, partiamo dal nodo iniziale e seguiamo per i primi nodi quelli dettati dal percorso dei predecessori e poi seguiamo il percorso che vogliamo fare noi, in questo caso ciclo; come si vede, esiste un vertice che era già presente nel cammino e che, idealmente, lo migliora ad iterazioni.

Risposte con limiti di archi (al più 5 archi)

e) È possibile disegnare albero e grafo dei cammini minimi? No, in quanto abbiamo etichette instabili (no soluzione ammissibile duale e quindi no cammini minimi). Fermandosi all’iterazione 5, vedo che è stato aggiornato e, seguendo da questo la catena dei predecessori, arrivo a costruire l’albero dei cammini minimi.

Senza fare calcoli, non posso sapere se le etichette si stabilizzeranno e quindi dato che non faccio l’iterazione 6, non riesco a fare il grafo con cammini minimi.