Esercitazione 9

corso di laurea in **Matematica**

Informatica Generale, Lezione **24.4**

Ivano Salvo

SAPIENZA
UNIVERSITÀ DI ROMA

Esercitazione 9, Lezione 24

1. Supponete di avere un grafo G con pesi sugli archi w e una soluzione dei cammini minimi da sorgente unica s sottoforma di vettore $c[u]$ dei costi dei cammini per arrivare a u e vettore dei padri p.

Fornire un algoritmo per **verificare che la soluzione sia corretta**.

2. Supponete di avere un un grafo G con pesi sugli archi w, **anche negativi**. Calcolate $W = \min_{e \in E} w(e)$ e sostituite tutti i pesi $w(e)$ con $w'(e) = w(e) + |W|$.

L'algoritmo di Dijkstra usando w' come pesi, restituisce risultati corretti rispetto al problema originario? Fornire un controesempio. Quali cammini vengono "privilegiati"?

3. Dato un grafo diretto G, fornire un algoritmo per determinare se G sia semi-connesso o meno [SUGG: conviene prima ragionare sui DAG]

4. Dato un grafo diretto G, fornire un algoritmo per determinare un grafo G' che ha gli stessi nodi e le stesse componenti connesse di G, ma un **numero minimo archi**.

Soluzioni Esercitazione 9

corso di laurea in **Matematica**

Informatica Generale, Lezione **25.2**

Ivano Salvo

SAPIENZA
UNIVERSITÀ DI ROMA

Esercizio 1 del 27/5/2022

⏵ **Esercizio 1**: Supponete di avere un grafo G con pesi sugli archi w e una soluzione dei cammini minimi da sorgente unica s sottoforma di vettore $c[u]$ dei costi dei cammini per arrivare a u e vettore dei padri p.

Fornire un algoritmo per **verificare che la soluzione sia corretta**.

Un modo per verificarla è la seguente:

1. verificare che il **vettore dei padri p** e **vettori dei costi c sono "fedeli"**, quindi dev'essere $c(p[u]) + w(u, v) = c(v)$ e, ovviamente, dev'esserci l'arco (u, v)
2. verificare che **non ci sono possibilità di "migliorare"** i costi seguendo altri cammini. Un modo è **fare un "giro di Bellman-Ford"** e verificare che `relax` non modifica nessun costo minimo.

Il costo di **1.** è $\Theta(n)$, mentre il costo di **2.** necessita di verificare tutti gli archi.

Dipende dalla rappresentazione, può essere $\Theta(m)$ [a lista di archi], $\Theta(n+m)$ [con liste di adiacenza], $\Theta(n^2)$ [con matrice di adiacenza]

Esercizio 2 del 27/5/2022

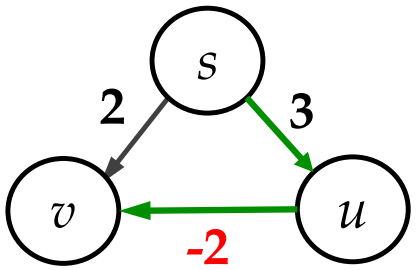
⏵ **Esercizio 2**: Supponete di avere un un grafo G con pesi sugli archi w, **anche negativi**. Calcolate $W = \min_{e \in E} w(e)$ e sostituite tutti i pesi $w(e)$ con $w'(e) = w(e) + |W|$.

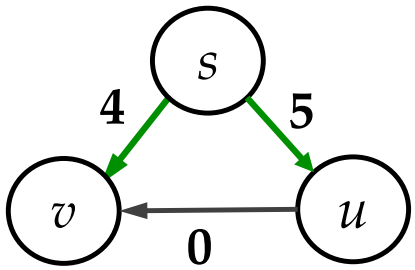
L'algoritmo di Dijkstra usando w' come pesi, restituisce risultati corretti rispetto al problema originario? Fornire un controesempio. Quali cammini vengono "privilegiati"?

La risposta è **NO**.

Intuitivamente, il problema è che un cammino lungo n "peggiora" il suo peso di $n|W|$ e quindi vengono favoriti i cammini più corti.

Un immediato controesempio si vede con 3 nodi, che è anche un **controesempio** al fatto che Dijkstra possa funzionare con **archi di peso negativo**.





Esercizio 3 del 27/5/2022

Esercizio 3: Dato un grafo diretto G, fornire un algoritmo per determinare se G sia semi-connesso o meno [SUGG: conviene prima ragionare sui DAG]

Prendiamo **due nodi consecutivi** nell'ordine topologico in un DAG G semi-connesso, v_i e v_{i+1}: **c'è un arco $v_i \to v_{i+1}$**?

Se non ci fosse, non è possibile né $v_i \rightsquigarrow v_{i+1}$ (dovrei passare o per nodi v_j con $j < i$ o per nodi v_k con $k > i+1$), né $v_{i+1} \rightsquigarrow v_i$ perché avrei un cammino che "risale" sull'ordine topologico.

Quindi, un DAG semi-connesso, **contiene necessariamente** il cammino $v_0 \to v_1 \to \ldots \to v_i \to v_{i+1} \to \ldots v_{n-1} \to v_n$.

Osservando che la connessione è totale dentro le componenti fortemente connesse, preso un grafo diretto qualsiasi G:

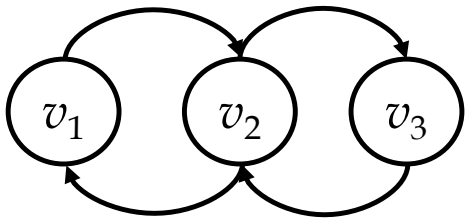
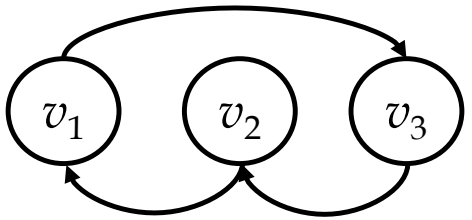
1. prima si calcolano le **SCC** (costo $\Theta(n + m)$),
2. poi il **DAG condensato G^{SCC}** (costo $\Theta(n + m)$)
3. si calcola il **topological sort su G^{SCC}** (costo $\Theta(n + m)$), e
4. si verifica che ci sia un arco tra ogni nodo e il suo successore in G^{SCC} (costo $\Theta(n + m)$).

Le complessità sono date con liste di adiacenza.

Esercizio 4 del 27/5/2022

⏵ **Esercizio 4**: Dato un grafo diretto G, fornire un algoritmo per determinare un grafo G' che ha gli stessi nodi e le stesse componenti connesse di G, ma un **numero minimo archi**.

Il modo più economico (in termini di archi) per avere n nodi in una componente connessa è avere un unico ciclo semplice con n archi.

Osserviamo che questo non è necessariamente un sottografo (nell'esempio a sinistra, nessun sottografo $G'=(\{v_1, v_2, v_3\}, E')$ è una componente fortemente connessa.

In un DAG, il grafo minimo che mantiene la stessa connessione è la riduzione transitiva (se $u \to v$, $v \to z$ allora posso omettere l'eventuale arco $u \to z$). ⏵ **Esercizio**: calcolare la riduzione transitiva.