Appello del 16 Giugno 2008
Soluzioni

Esercizio 1. Si consideri il seguente pseudocodice:

```
    algoritmo CamminiMinimi(grafo G, nodo s) → distanze
1.    for each ( nodo u in G ) do D_su ← +∞
2.    D_ss ← 0
3.    struttura dati R
4.    R.aggiungi(s)
5.    while ( not R.isEmpty() ) do
6.       u ←R.estrai()
7.       for each ( arco (u, v) in G ) do
8.          if (D_su + w(u, v) < D_sv) then
9.             D_sv ← D_su + w(u, v)
10.            if ( v ∉ R ) then R.aggiungi(v)
11.   return D
```

G è un grafo con pesi arbitrari sugli archi, dati dalla funzione peso w, ma senza cicli negativi. Assumendo che la struttura dati R implementi una coda (`aggiungi=enqueue`, `estrai=dequeue`):

1. Dimostrare che al termine dell'esecuzione, per ogni nodo u, $D_{su} = d_{su}$, ovvero D_{su} rappresenta la distanza corretta del nodo u dalla sorgente s.

   [*Suggerimento*: si consideri l'esecuzione concettualmente divisa in passate, dove nell'i-esima passata sono estratti dalla coda i nodi inseriti nella $(i-1)$-esima passata. Si pensi poi alla dimostrazione di correttezza dell'algoritmo di Bellman, Ford e Moore.]

   Soluzione: sia $\pi^*_{sv_k} = \langle s, v_1, \ldots, v_k \rangle$ un cammino minimo inizialmente ignoto di cui vogliamo trovare il costo d_{sv_k}. Consideriamo l'esecuzione dell'algoritmo. Originariamente, solo la sorgente s è in coda. Quando s viene estratta nella prima passata, tutti i suoi vicini sono accodati: questo insieme sicuramente contiene anche il nodo v_1, e quindi l'arco (s, v_1) viene correttamente rilassato nella prima passata. Alla seconda passata tutti i vicini di s, uno ad uno, sono estratti dalla coda e gli archi ad essi incidenti sono rilassati: tra questi archi, troveremo sicuramente anche l'arco (v_1, v_2). Inoltre, le destinazioni degli archi considerati (tra cui il nodo v_2) sono inserite in coda (se non vi compaiono già), per essere estratte durante la terza passata, che rilasserà certamente l'arco (v_2, v_3). Osserviamo che se v_2 non viene reinserito in coda, allora vuol dire che è già presente (ovvero che è stato inserito durante la prima passata), e in questo caso l'arco (v_2, v_3) viene rilassato durante la seconda passata. In entrambi i casi, entro il termine della terza passata il secondo arco di $\pi^*_{sv_k}$ sarà stato correttamente rilassato. Come nel caso di Bellman, Ford e Moore, si procede in questo modo per al più n passate.

   L'osservazione cruciale alla base della correttezza dell'algoritmo `CamminiMinimi` può quindi essere sintetizzata come segue. Consideriamo un nodo u tale che D_{su} non cambia durante una passata. Nella passata successiva è inutile considerare gli archi (u, v) uscenti da u: risulterà infatti $D_{sv} \leq D_{su} + w(u, v)$, poiché la condizione era vera precedentemente e D_{su} è immutata. Non occorre quindi verificare la condizione di Bellman nuovamente. Non è difficile vedere che valgono le stesse proprietà discusse per l'algoritmo di Bellman, Ford e Moore: entro la i-esima passata avremo rilassato un insieme di archi che certamente include (s, v_1), (v_1, v_2), ... (v_{i-1}, v_i), e dopo n passate la coda sarà vuota e le stime delle distanze risulteranno esatte.

2. Mettere in relazione i rilassamenti eseguiti dall'algoritmo `CamminiMinimi` con quelli eseguiti dall'algoritmo di Bellman, Ford e Moore: si tratta degli stessi rilassamenti, di un sovrainsieme, di un sottoinsieme, o nessuna delle precedenti? Perché?

Soluzione: ricordiamo innanzitutto che l'algoritmo di Bellman, Ford e Moore esegue al più n passate, e in ogni passata rilassa *tutti* gli archi del grafo. Consideriamo ora l'algoritmo `CamminiMinimi`, con la suddivisione in passate introdotta nel punto 1 di questo esercizio. Nella prima passata la coda contiene solo la sorgente s, e dopo l'estrazione di s l'algoritmo rilassa solo gli archi del tipo (s, x), inserendo x in coda: l'insieme dei rilassamenti eseguiti nella prima passata è quindi un sottoinsieme di quelli eseguiti da Bellman, Ford e Moore. All'inizio della seconda passata la coda contiene i vicini della sorgente s, e l'algoritmo estrae questi nodi uno ad uno. Sia x un tale vicino: l'algoritmo rilassa solo gli archi del tipo (x, y), inserendo eventualmente y in coda. Anche l'insieme dei rilassamenti eseguiti nella seconda passata è quindi un sottoinsieme di quelli eseguiti da Bellman, Ford e Moore. Lo stesso ragionamento vale per tutte le passate successive. Poiché nel punto 1 abbiamo dimostrato che le passate sono al più n come nel caso di Bellman, Ford e Moore, risulta che i rilassamenti eseguiti dall'algoritmo `CamminiMinimi` sono nel complesso un sottoinsieme dei rilassamenti eseguiti dall'algoritmo di Bellman, Ford e Moore. Al più, i due insiemi coincidono.

Un ragionamento alternativo per rispondere (parzialmente) alla domanda è il seguente: se esiste almeno un nodo x non raggiungibile dalla sorgente s avente un arco uscente (x, y), x non viene mai inserito in coda e quindi l'arco (x, y) non sarà mai rilassato dall'algoritmo `CamminiMinimi`. Questo implica che potrebbero esserci degli archi che Bellman, Ford e Moore rilassa e l'algoritmo `CamminiMinimi` non rilassa. Quindi, poiché Bellman, Ford e Moore rilassa ad ogni passata tutti gli archi, si ha che i rilassamenti eseguiti dall'algoritmo `CamminiMinimi` sono in generale un sottoinsieme dei rilassamenti eseguiti dall'algoritmo di Bellman, Ford e Moore. Questo ragionamento, seppur corretto, non è però del tutto soddisfacente, perché si applica solo a particolari tipi di istanze, quelle in cui esiste almeno un nodo non raggiungibile dalla sorgente. Esistono invece anche grafi in cui tutti i nodi sono raggiungibili da s, ma i rilassamenti eseguiti dall'algoritmo `CamminiMinimi` sono comunque un sottoinsieme proprio dei rilassamenti eseguiti dall'algoritmo di Bellman, Ford e Moore (vedi punto 4 dell'esercizio). Il primo ragionamento copre anche questi casi.

3. Dimostrare che il tempo di esecuzione dell'algoritmo `CamminiMinimi` è $O(m\,n)$, dove n ed m sono rispettivamente il numero di nodi e di archi del grafo.

   Soluzione: non è difficile convincersi che il tempo di esecuzione è dominato dalle operazioni di rilassamento (test alla riga 8 del codice). Abbiamo visto nel punto precedente che l'algoritmo `CamminiMinimi` rilassa un sottoinsieme degli archi rilassati da Bellman, Ford e Moore. Poiché quest'ultimo algoritmo ha tempo di esecuzione $O(n \cdot m)$, anche il tempo di esecuzione dell'algoritmo `CamminiMinimi` deve quindi essere $O(n \cdot m)$.

   Il seguente ragionamento non è invece corretto. Il while della riga 5 viene eseguito al più n volte, perché ciascun nodo (raggiungibile da s) viene inserito in coda una volta. Il for della riga 7 richiede tempo $O(m)$. Quindi il tempo di esecuzione è $O(n \cdot m)$. Ci sono due errori, di cui il primo molto grave:

   - non si tiene conto che un nodo potrebbe essere inserito in coda più di una volta;
   - il for della riga 7 richiede tempo $O(n)$ perché esamina solo gli archi uscenti da un nodo fissato (quindi questo ragionamento porterebbe a una stima $O(n^2)$, che però è errata).

4. Mostrare un'istanza in cui il tempo di esecuzione dell'algoritmo `CamminiMinimi` è asintoticamente inferiore a quello dell'algoritmo di Bellman, Ford e Moore.

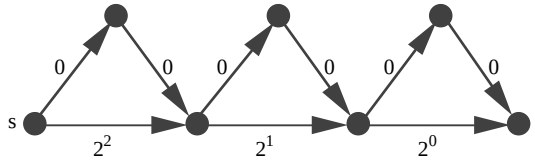
   Soluzione: Basta considerare una catena di n nodi, $v_1\ v_2\ ...\ v_{n-1}\ v_n$, con sorgente $s = v_1$. Ad ogni passata l'algoritmo di Bellman, Ford e Moore rilassa tutti gli $n - 1$ archi, per un tempo di esecuzione totale $\Theta(n^2)$. L'algoritmo `CamminiMinimi`, invece, rilassa un solo arco (quello uscente dall'unico nodo in coda), e richiede tempo $\Theta(1)$ per passata. Il tempo di esecuzione totale è quindi $\Theta(n)$.

   Osserviamo che nella soluzione viene mostrata un'istanza *parametrica nel numero di nodi* n, e quindi una famiglia di istanze: le catene di 2 nodi, di 3 nodi, di 4 nodi, e in generale di n nodi. Questo perché l'esercizio richiede di mostrare un caso in cui il tempo di esecuzione dell'algoritmo `CamminiMinimi` sia *asintoticamente* inferiore a quello dell'algoritmo di Bellman, Ford e Moore: il ragionamento deve quindi valere per grafi

di dimensione arbitrariamente grande. Per rispondere alla domanda non basta mostrare un grafo con un numero costante di nodi (ad esempio, 3 o 4, come in molte delle soluzioni tipicamente proposte dagli studenti), su cui l'algoritmo `CamminiMinimi` esegue meno rilassamenti di Bellman, Ford e Moore: il tempo di esecuzione di entrambi gli algoritmi su grafi di dimensione costante è comunque $O(1)$!

Assumendo che la struttura dati R implementi una pila (`aggiungi=push`, `estrai=pop`):

5. Mostrare l'esecuzione dell'algoritmo sul seguente grafo:



assumendo che nella linea 7 del codice l'arco uscente da u ed avente peso 0 sia sempre considerato per primo.

Soluzione: chiamiamo 1, 3, 5 e 7 i quattro nodi inferiori (da sinistra verso destra), e 2, 4, 6 i tre nodi superiori (sempre da sinistra verso destra). Mostriamo, ad ogni passo, il conntenuto della pila R e le stime delle distanze da $s = 1$. $D_{s1} = 0$ durante tutta l'esecuzione e non lo mostreremo. (In alcuni casi, mostreremo anche una stringa σ il cui significato sarà chiarito nel punto successivo dell'esercizio.)

Dopo l'inizializzazione (righe 1–4 del codice):

1

$D_{s2} = +\infty$
$D_{s3} = +\infty$
$D_{s4} = +\infty$
$D_{s5} = +\infty$
$D_{s6} = +\infty$
$D_{s7} = +\infty$

Dopo l'estrazione del nodo 1 e l'esecuzione delle istruzioni nelle righe 7–10:

3
2

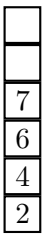
$D_{s2} = 0$
$D_{s3} = 2^2$
$D_{s4} = +\infty$
$D_{s5} = +\infty$
$D_{s6} = +\infty$
$D_{s7} = +\infty$

Dopo l'estrazione del nodo 3 e l'esecuzione delle istruzioni nelle righe 7–10:

5
4
2

$D_{s2} = 0$
$D_{s3} = 2^2$
$D_{s4} = 2^2$
$D_{s5} = 2^2 + 2^1$
$D_{s6} = +\infty$
$D_{s7} = +\infty$

Dopo l'estrazione del nodo 5 e l'esecuzione delle istruzioni nelle righe 7–10:

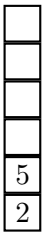


$D_{s2} = 0$
$D_{s3} = 2^2$
$D_{s4} = 2^2$
$D_{s5} = 2^2 + 2^1$
$D_{s6} = 2^2 + 2^1$
$D_{s7} = 2^2 + 2^1 + 2^0$ $\sigma = 111$

Dopo l'estrazione del nodo 7 le distanze non variano. Dopo la successiva estrazione del nodo 6 e l'esecuzione delle istruzioni nelle righe 7–10:

$D_{s2} = 0$
$D_{s3} = 2^2$
$D_{s4} = 2^2$
$D_{s5} = 2^2 + 2^1$
$D_{s6} = 2^2 + 2^1$
$D_{s7} = 2^2 + 2^1$ $\sigma = 110$

Dopo l'estrazione del nodo 7 le distanze non variano. Dopo la successiva estrazione del nodo 4 e l'esecuzione delle istruzioni nelle righe 7–10:

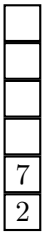


$D_{s2} = 0$
$D_{s3} = 2^2$
$D_{s4} = 2^2$
$D_{s5} = 2^2$
$D_{s6} = 2^2 + 2^1$
$D_{s7} = 2^2 + 2^1$

Dopo l'estrazione del nodo 5 e l'esecuzione delle istruzioni nelle righe 7–10:

$D_{s2} = 0$
$D_{s3} = 2^2$
$D_{s4} = 2^2$
$D_{s5} = 2^2$
$D_{s6} = 2^2$
$D_{s7} = 2^2 + 2^0$ $\sigma = 101$

Dopo l'estrazione del nodo 7 le distanze non variano. Dopo la successiva estrazione del nodo 6 e l'esecuzione delle istruzioni nelle righe 7–10:



$D_{s2} = 0$
$D_{s3} = 2^2$
$D_{s4} = 2^2$
$D_{s5} = 2^2$
$D_{s6} = 2^2$
$D_{s7} = 2^2$ $\sigma = 100$

Dopo l'estrazione del nodo 7 le distanze non variano. Dopo la successiva estrazione del nodo 2 e l'esecuzione delle istruzioni nelle righe 7–10:

$D_{s2} = 0$
$D_{s3} = 0$
$D_{s4} = 2^2$
$D_{s5} = 2^2$
$D_{s6} = 2^2$
$D_{s7} = 2^2$

Dopo l'estrazione del nodo 3 e l'esecuzione delle istruzioni nelle righe 7–10:

$D_{s2} = 0$
$D_{s3} = 0$
$D_{s4} = 0$
$D_{s5} = 2^1$
$D_{s6} = 2^2$
$D_{s7} = 2^2$

Dopo l'estrazione del nodo 5 e l'esecuzione delle istruzioni nelle righe 7–10:

$D_{s2} = 0$
$D_{s3} = 0$
$D_{s4} = 0$
$D_{s5} = 2^1$
$D_{s6} = 2^1$
$D_{s7} = 2^1 + 2^0$ $\qquad\qquad$ $\sigma = 011$

Dopo l'estrazione del nodo 7 le distanze non variano. Dopo la successiva estrazione del nodo 6 e l'esecuzione delle istruzioni nelle righe 7–10:

$D_{s2} = 0$
$D_{s3} = 0$
$D_{s4} = 0$
$D_{s5} = 2^1$
$D_{s6} = 2^1$
$D_{s7} = 2^1$ $\qquad\qquad$ $\sigma = 010$

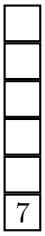
Dopo l'estrazione del nodo 7 le distanze non variano. Dopo la successiva estrazione del nodo 4 e l'esecuzione delle istruzioni nelle righe 7–10:

$D_{s2} = 0$
$D_{s3} = 0$
$D_{s4} = 0$
$D_{s5} = 0$
$D_{s6} = 2^1$
$D_{s7} = 2^1$

Dopo l'estrazione del nodo 5 e l'esecuzione delle istruzioni nelle righe 7–10:

$D_{s2} = 0$
$D_{s3} = 0$
$D_{s4} = 0$
$D_{s5} = 0$
$D_{s6} = 0$
$D_{s7} = 2^0$ $\sigma = 001$

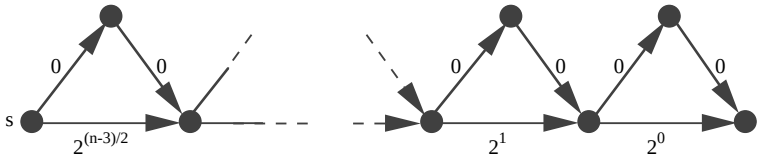
Dopo l'estrazione del nodo 7 le distanze non variano. Dopo la successiva estrazione del nodo 6 e l'esecuzione delle istruzioni nelle righe 7–10:



$D_{s2} = 0$
$D_{s3} = 0$
$D_{s4} = 0$
$D_{s5} = 0$
$D_{s6} = 0$
$D_{s7} = 0$ $\sigma = 000$

Infine viene estratto il nodo 7 e l'esecuzione termina essendo la pila vuota.

6. Generalizzare il ragionamento a grafi del tipo:



Qual è il tempo di esecuzione in funzione di n? Si assuma di poter eseguire operazioni aritmentiche con precisione arbitraria in tempo $O(1)$.

Soluzione : consideriamo innazitutto, per chiarirci le idee, l'istanza discussa nel punto 5. Soffermiamoci sul nodo 7, che è apparentemente il più distante dalla sorgente. D_{s7} decresce durante l'esecuzione da $+\infty$ fino a 0. Consideriamo gli istanti durante l'esecuzione in cui D_{s7} decresce: in tutti e soli questi istanti abbiamo mostrato una stringa σ il cui significato sarà ora chiarito.

Ciascun nodo i dispari (i nodi del livello inferiore nel disegno) puó essere raggiunto dal nodo dispari immediatamente precedente, il nodo $i-2$, in due modi: sfruttando l'arco orizzontale $(i-2, i)$ di peso > 0, o sfruttando i due archi diagonali $(i-2, i-1)$ e $(i-1, i)$ di peso 0. Rappresenteremo la prima scelta con un numero 1 nella stringa σ, e la seconda scelta con un numero 0.

La stringa σ è una codifica binaria del miglior cammino usato in ogni istante per raggiungere il nodo 7: ciascun bit (dal più significativo a quello meno significativo) rappresenta la scelta del cammino orizzontale o del cammino fatto da due archi diagonali per passare da un nodo dispari a quello successivo (considerando i nodi dispari da sinistra a destra). Ad esempio, $\sigma = 101$ codifica il cammino 1 3 4 5 7: il primo bit pari a 1 identifica il sottocammino orizzontale $(1, 3)$, il secondo bit pari a 0 identifica il sottocammino diagonale composto dagli archi $(3, 4)$ e $(4, 5)$, il terzo bit pari a 1 identifica il sottocammino orizzontale $(5, 7)$.

Osserviamo che la stringa σ decresce da 111 a 000 e si comporta come un contatore binario. Vengono eseguiti quindi 2^3 decrementi, essendo composta da 3 bit.

In generale, su istanze di n nodi, la stringa σ è composta da $(n-1)/2$ bit, e quindi il numero di decrementi di D_{sn} eseguiti dall'algoritmo è $2^{(n-1)/2}$. Il tempo di esecuzione è pertanto $\Omega(2^{(n-1)/2})$. Con un ragionemento più elaborato si può dimostrare che il tempo di esecuzione è $\Theta(2^{(n-1)/2})$. Ai fini dell'esercizio, bastava

comunque capire che l'algoritmo che usa una pila è molto inefficiente e il tempo di esecuzione è almeno esponenziale. Il cattivo comportamento è dovuto al fatto che, nella scansione degli archi uscenti dai nodi dispari, abbiamo sempre considerato prima l'arco uscente diagonale e poi quello orizzontale. Il successivo nodo dipari si ritroverà quindi sul top della pila, portando prima a considerare i cammini orizzontali, meno convenienti perché di peso maggiore, sostituendoli via via con i corrispondenti cammini diagonali di peso 0. I rilassamenti vengono quindi eseguiti nel peggior ordine possibile.

◁ ◇ ▷

Esercizio 2. Si analizzino correttezza e tempo di esecuzione dell'algoritmo di capacity scaling.

Soluzione. Vedere il contenuto del paragrafo 7.3 del libro di testo *Network flows* (Ahuja, Magnanti, Orlin).