Algoritmi per Bioinformatica **Anno Acc. 2014-15**

Esercizi 2

1. Vogliamo analizzare una rete sociale rispetto ad una nuova misura di vicinanza che ci servirà per stimare il grado di conoscenza di due partecipanti. La rete sociale è rappresentata da un grafo non orientato in cui i nodi rappresentano gli individui partecipanti ed un arco tra i nodi u e v indica che u e v sono in contatto.

   La nostra misura di vicinanza si basa sul seguente ragionamento: due persone potrebbero avere un conoscente in comune per caso pur non essendo affatto vicine (in termini di intensità di relazione). Tuttavia se due persone hanno molti conoscenti in comune, probabilmente si incontreranno spesso e quindi in tal caso le reputiamo vicine.

   Più in generale, quindi, non siamo disposti a concludere che u e v sono vicini semplicemente perché la loro "distanza" nel grafo è piccola, ma solo se esistono numerosi esempi di cammini tra u e v di lunghezza "piccola".

   Quindi, vogliamo un algoritmo che dato un grafo $G = (V, E)$ e due vertici u e v ci dica "rapidamente" (i) il minimo numero di archi che dobbiamo attraversare per andare da u a v, (ii) quanti sono i cammini da u a v di lunghezza minima, cioè il numero totale di cammini la cui lunghezza è pari al minimo numero di archi per andare da u a v.

2. Supponiamo che la banca centrale europea riformi l'Euro e decida di utilizzare come monete solo tagli da 1, 5, 10, 20 e 25 centesimi. L'algoritmo del cassiere continuerebbe ad essere ottimale? Se sì, si fornisca una prova. Se no, si fornisca un controesempio e si dica se aggiungendo od eliminando un solo tipo di moneta, sia possibile recuperare l'ottimalità delll'algoritmo del cassiere.

3. Si analizzi la complessità dell'algoritmo del cassiere visto a lezione per la soluzione del problema del cambia-monete. Per la stima della complessità, si consideri come taglia dell'input $n + log_2 S$.

4. Supponiamo di avere un grafo connesso G che rappresenta i collegamenti aerei tra varie regioni. Ogni nodo rappresenta una regione e ogni arco (u, v) indica la possibilità di andare dalla regione u alla regione v. Diciamo che una regione z è **critica** se, in caso di cancellazione di tutti i suoi collegamenti, esistono almeno altre due regioni u e v tali che non sarà possibile più andare da u a v.

   Formalizziamo il problema nel seguente modo: Sia $G = (V, E)$ un grafo non orientato. Diciamo che un nodo $v \in V$ è un **nodo critico** se rimuovendo v da G, il grafo risultante $G - v$ non è più connesso. Ricordiamo che $G - v$ è il grafo il cui insieme di vertici è dato da tutti i vertici di V tranne il vertice v; e l'insieme degli archi di $G - v$ è dato da tutti gli archi di E che non incidono sul nodo v.

   Vogliamo quindi un algoritmo che dato un grafo $G = (V, E)$ dia in output tutti i nodi critici.

5. Consideriamo un grafo orientato aciclico G. Supponiamo di eseguire una visita DFS su G e di fornire in output la lista $v_1, v_2, \ldots, v_n$ dove v_i è l' i-esimo vertice di G visitato dalla DFS.

   Possiamo affermare che $v_1, v_2, \ldots, v_n$ costituisce un ordinamento topologico di G?

   Se sì, provare a dimostrarlo; se no, fornire un controesempio: cioè un esempio di grafo aciclico orientato ed una sua vista DFS tale che la lista di vertici nell'ordine in cui vengono visitati dalla DFS non rappresenta un ordinamento topologico.

6. Abbiamo un nuovo tutor per il corso di algoritmi che sembra estremamente severo e gira anche voce che si diverta a torturare gli studenti.

   Durante la prima lezione ha assegnato n esercizi, $e_1, \ldots e_n$. Per ogni esercizio e_i ha fornito il tempo medio necessario per risolverlo t_i e la scadenza d_i entro cui l'esercizio risolto andrà consegnato.

   Ammetterà che alcuni esercizi possano essere consegnati in ritardo rispetto alla scadenza. Ma ha indicato il massimo ritardo ammissibile T, sul quale non è disposto a transigere. Quindi se anche uno solo degli esercizi venisse consegnato con un ritardo superiore a T rispetto alla sua scadenza non si verrà ammessi all'esame.

   Sappiamo di poter lavorare al massimo su un solo esercizio alla volta e di riuscire a risolverlo nel tempo medio indicato. Quindi se, per esempio ci sono solo 3 esercizi e decidiamo di affrontare prima e_2 poi e_1 e poi e_3 consegneremo e_2 al tempo t_2; quindi consegneremo e_1 al tempo $t_2 + t_1$ ed infine consegneremo e_3 al tempo $t_2 + t_1 + t_3$.

   E quindi se $t_2 > d_2 + T$ oppure $t_2 + t_1 > d_1 + T$ oppure $t_2 + t_1 + t_3 > d_3 + T$ non saremo ammessi all'esame.

   Sapendo che, oltre che severo, il tutor si diverte a torturarci, prima di metterci a lavorare, vogliamo sapere se è possibile organizzarci in modo da soddisfare i vincoli di cui sopra. Una volta stabilito che ciò è possibile e avendo trovato l'ordine in cui risolvere gli esercizi saremmo sicuri dell'ammissione. Se invece scoprissimo che non è possibile soddisfare i vincoli posti dal tutor, avremmo la prova della sua malvagità e potremmo ricorrere al responsabile della didattica e richiedere la sua espulsione.

   Vogliamo dunque progettare un algoritmo, efficiente, che avendo in input i tempi $t_1, \ldots, t_n$, le scadenze $d_1, \ldots, d_n$ e la massima tolleranza T per il ritardo, ci permetta di sapere se esiste un ordinamento degli esercizi tale che, risolvendoli in quell'ordine, consegneremmo ogni esercizio e_i entro $d_i + T$ cioè con un ritardo non superiore a T rispetto alla sua scadenza. Inoltre il nostro algoritmo deve essere in grado di fornirci tale ordinamento degli esercizi se esso esiste.

   Esempio 1. *Tempi*: $t_1 = 7, t_2 = 3, t_3 = 10, t_4 = 1$; *Scadenze*: $d_1 = 7, d_2 = 10, d_3 = 17, d_4 = 18$; *Tolleranza*: $T = 5$

   Possibile Soluzione: e_2, e_1, e_4, e_3

   Con questo ordinamento, e_2 viene consegnato al tempo 3 senza ritardo; e_1 viene consegnato al tempo 10 senza ritardo; e_4 è consegnato al tempo 11 senza ritardo ed e_3 viene consegnato al tempo 21 con un ritardo pari a 4 rispetto alla scadenza $d_3 = 17$. Quindi nessuna consegna avviene con ritardo > 5.

   Esempio 2 *Tempi*: $t_1 = 5, t_2 = 15, t_3 = 10$; *Scadenze*: $d_1 = 5, d_2 = 20, d_3 = 15$; *Tolleranza*: $T = 8$

   Si può vedere che in questo caso qualsiasi ordinamento porta ad almeno una consegna con un ritardo superiore ad 8. Quindi non c'è soluzione.

7. Fornire un algoritmo che dato un grafo non orientato G, verifica se G contiene o meno un ciclo. Se G contiene un ciclo, l'algoritmo deve dare in output un ciclo di G. Se G non contiene alcun ciclo, l'algoritmo deve dare in output la stringa "nessun ciclo".

8. Un gruppo di 4 persone $\{a, b, c, d\}$ con una sola torcia deve attraversare un ponte cadente. Al più 2 persone per volta possono attraversare il ponte. Ogni attraversamento deve essere fatto portando la torcia. Quindi la torcia deve viaggiare avanti e indietro.

   Le quattro persone hanno velocità differenti ed i tempi di attraversamento per ognuno di essi sono: $t(a) = 1, t(b) = 2, t(c) = 5, t(d) = 10$. Quando due persone attraversano insieme il ponte lo fanno alla velocità del più lento tra di loro. Quindi se a e b fanno un viaggio insieme, esso durerà 2, se c e d viaggiano insieme attraverseranno il ponte in 10 untià di tempo.

   Definire una strategia per minimizzare il tempo totale necessario per portare tutte le persone dall'altra parte del ponte. Provare a dimostrare che tale strategia è ottimale, nel senso che non esiste un'altra sequenza di attraversamenti che porta le persone dall'altra parte del ponte in un tempo inferiore.