Esercizi a lezione

docente: Luciano Gualà

Di seguito si trovano alcuni degli esercizi che sono stati discussi durante il corso (oltre quelli già presenti nelle slide e quelli dei Problem Set). Lo studente che vuole avere una preparazione profonda per la prova scritta è caldamente invitato a rivedere (o, meglio, studiare) le soluzioni proposte a lezione per questi esercizi.

Esercizio 1 (*notazioni asintotiche, costanti ed esponenti*)
Siano $f(n)$ e $g(n)$ due funzioni sempre positive. Si dimostri o si confuti la seguente relazione: $f(n) = O(g(n))$ implica $2^{f(n)} = O(2^{g(n)})$.

Esercizio 2
Sia $A[1;n]$ un array di n interi distinti ordinato in modo crescente. Progettare un algoritmo con complessità temporale $o(n^2)$ che, preso in input A e un valore x, dice se esistono due indici i e j tale che $A[i] + A[j] = x$.

Esercizio 3
Sia $A[1;n]$ un vettore di n numeri. Si progetti un algoritmo che restituisca due indici i^* e j^*, con $i^* < j^*$, che massimizzano $A[j^*] - A[i^*]$, ovvero due indici tale che $A[j^*] - A[i^*] \geq A[j] - A[i]$, per ogni i, j con $i < j$. L'algoritmo deve impiegare tempo $O(n)$.

Esercizio 4
Sia T un albero binario di n nodi in cui ad ogni nodo v è associato un valore reale positivo $k(v)$ e un colore $c(v) \in \{rosso, nero\}$. Diciamo che un cammino da un nodo v alla radice è rosso se tutti i nodi lungo il cammino sono di colore rosso; inoltre definiamo il *valore* di un tale cammino come la somma dei valori dei nodi del cammino. Progettare un algoritmo che, dato T, restituisce il valore del cammino rosso di tipo nodo-radice di valore massimo. L'algoritmo deve avere complessità temporale $O(n)$. Si assuma che l'albero è mantenuto attraverso una struttura dati collegata e che per ogni nodo siano disponibili, oltre alla chiave e al colore, i puntatori al padre e ai due figli.

Esercizio 5
Sia T un albero binario di n nodi. La profondità di un nodo v è la lunghezza (misurata in termini di numero di archi) del cammino da v alla radice. Progettare un algoritmo che, dato T e un intero h, restituisce il numero di nodi di T che hanno profondità almeno h. L'algoritmo deve avere complessità temporale $O(n)$. Si assuma che l'albero è mantenuto attraverso una struttura dati collegata e che per ogni nodo siano disponibili i puntatori al padre e ai due figli.

Esercizio 6
Sia T un albero binario di n nodi in cui ad ogni nodo v è associato un valore reale positivo $k(v)$. Diciamo che un nodo v è *bilanciato* se la somma dei valori degli antenati di v è uguale alla somma dei valori dei discendenti di v. Progettare un algoritmo che, dato T, restituisce

il numero di nodi bilanciati di T. L'algoritmo deve avere complessità temporale $O(n)$. Si assuma che l'albero è mantenuto attraverso una struttura dati collegata e che per ogni nodo siano disponibili i puntatori al padre e ai due figli.

Esercizio 7
Sia $V[1 : n]$ un vettore di n caratteri, dove ogni posizione può contenere un carattere nell'insieme $\{Y, E, S\}$. Il vettore è organizzato in modo che, se letto da sinistra a destra, si ottiene prima una sequenza non vuota di Y, poi una sequenza non vuota di E, e poi una sequenza non vuota di S. Si progetti un algoritmo con complessità $o(n)$ che calcoli il numero di Y, di E e di S contenute nel vettore. Si fornisca lo pseudocodice dell'algoritmo.

Esercizio 8 (*calcolo della maggioranza*)[1]
Sia dato un vettore $V[1; n]$ di n elementi. Un elemento è di *maggioranza* se il numero di occorrenze dell'elemento è strettamente più di $n/2$. Si assuma che nel vettore c'è un elemento di maggioranza (che si vuole scoprire) ma gli elementi sono in qualche modo "nascosti", non possono essere letti direttamente. L'unica operazione possibile è fare una *richiesta* $q(i, j)$, che consiste nel chiedere se i due elementi in posizione i e j, rispettivamente, sono uguali o meno; ovvero, $q(i, j)$ restituisce `vero` se $V[i] = V[j]$, `falso` altrimenti. Le operazioni di richiesta sono costose e quindi se ne vogliono fare il meno possibile.

(a) Si progetti un algoritmo che usa la tecnica del *divide et impera* che trova l'elemento di maggioranza facendo $o(n^2)$ richieste.

(b) Si progetti un (miglior) algoritmo che fa un numero lineare di richieste e usa memoria costante.

Esercizio 9
Si consideri il problema di trovare, dato un vettore ordinato $A[1; n]$ di n bit, ovvero dove $A[i] \in \{0, 1\}$ per ogni i, il numero k di zeri presenti in A. Si progetti un algoritmo con complessità $O(\log n)$ e poi un miglior algoritmo con tempo $O(\log k)$.

Esercizio 10
Sia $A[1; n]$ un vettore di n bit, ovvero di zeri e di uni. Si progetti un algoritmo che restituisca un indice k tale che il numero di zeri prima di k, ovvero in $A[1; k]$, è uguale al numero di uni dopo k, ovvero in $A[k + 1; n]$. L'algoritmo deve impiegare tempo $O(n)$. E' possibile risolvere il problema con la stessa complessità computazionale e usando memoria ausiliaria costante?

Esercizio 11
La vostra città è modellata come un grafo diretto e pesato $G = (V, E, w)$, dove ad ogni arco $e \in E$ è associato un peso $w(e) \geq 0$ che rappresenta il costo per attraversare l'arco (strada) e. Voi siete in s e dovete andare ad una festa di compleanno che si tiene in t. Ma prima dovete passare a comprare un regalo. Per ogni nodo $v \in V$, sapete che c'è un negozio in v

[1] La soluzione di questo esercizio può essere trovata nelle soluzioni del problem set 2, aa 2016/17, esercizio 3.

e che se compraste il regalo lì vi costerebbe $c(v)$. Progettate un algoritmo (efficiente) che calcoli una strategia per raggiungere la festa muniti di regalo e che vi faccia spendere il meno possibile in termini di costo di trasporto più il costo del regalo.

Esercizio 12
Sia $G = (V, E)$ un grafo non orientato e non pesato, dove ogni arco $e \in E$ ha uno fra tre possibili colori $\{1, 2, 3\}$, e sia $U \subseteq V$ un sottoinsieme di nodi *speciali*. Su un nodo $s \in V$ è posizionata una pedina, inizialmente di colore 1. Il vostro obiettivo è quello di portare la pedina su un nodo *target* $t \in V$. Per far ciò potete usare le seguenti *mosse*:

- `sposta`: se la pedina si trova sul nodo $u \in V$, ha colore i, e c'è un arco $(u, v) \in E$ di colore i, allora potete spostare la pedina da u a v. Il colore della pedina resta i;
- `cambia colore`: se la pedina si trova su un nodo speciale $u \in U$, potete ricolorare la pedina del colore che volete. La pedina in questo caso, dopo la mossa, resta sul nodo u.

Progettate un algoritmo efficiente che trova, se ne esiste una, la sequenza più corta di mosse che risolve l'istanza del gioco.

Esercizio 13 *(Gualà e Clementi vanno a vedere la (magggica) Roma)*
Una città è modellata come un grafo diretto e pesato $G = (V, E, w)$, dove ad ogni arco $e \in E$ è associato un peso $w(e) \geq 0$ che rappresenta il costo, in termini di benzina consumata, per attraversare l'arco (strada) e. In questa città, i vostri docenti del corso di algoritmi, Gualà e Clementi, vogliono andare a vedere la partita della Roma allo stadio, che si trova nel nodo t. Loro sono rispettivamente nei nodi s_1 e s_2, e possiedono una macchina ciascuno. Volendo, possono incontrarsi in un nodo del grafo, parcheggiare una delle due macchine, e proseguire insieme. Ma di solito in questa città, parcheggiare costa. Per ogni nodo v, dunque, conoscono il costo $c(v)$ del parcheggio presente in v (si può assumere per semplicità che $c(s_1) = c(s_2) = c(t) = 0$). Progettate un algoritmo che in tempo $O(m + n \log n)$ calcoli la soluzione che Gualà e Clementi devono adottare per spendere complessivamente il meno possibile in termini di corso della benzina più costo del parcheggio.

E se il grafo G fosse diretto?

Esercizio 14 *(Cammini minimi in grafi con pesi su archi e nodi)*
Sia dato un grafo $G = (V, E)$ non diretto in cui ad ogni arco $(u, v) \in E$ è associato un peso $w(u, v) \geq 0$ e ad ogni nodo $v \in V$ è associato un costo $c(v) \geq 0$. Dato un cammino $P = \langle v_0, v_1, \ldots, v_k \rangle$ da v_0 e v_k, definiamo il *costo totale* di P la somma dei pesi degli archi e dei costi dei nodi di P, ovvero $\sum_{i=0}^{k-1} w(v_i, v_{i+1}) + \sum_{i=0}^{k} c(v_i)$.

Progettare un algoritmo che, dati due nodi s e t trova un cammino da s a t in G di costo totale minimo. Analizzare la correttezza e la complessità computazionale dell'algoritmo proposto.