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.