Prof. L. Margara **Scritto del 11/01/2019**

Algoritmi e Strutture Dati

Esercizio 1.[11 punti]
Sia V (vettore) un heap di n elementi. Fornire un algoritmo ottimo A di tipo *comparison sort* per ordinare V. Dimostrare che A è ottimo.

Traccia Soluzione esercizio 1.
Ordinare un Heap ha la stessa complessità computazionale di ordinare un vettore disordinato. Per dimostrarlo basta notare che presi $n/2$ numeri (qualsiasi) è sempre possibile costruire un heap che contiene quegli elementi nelle foglie (in qualsiasi ordine) e aggiungere altri $n/2$ elementi (opportunamente scelti) nei nodi interni. Quindi un algoritmo ottimo per ordinare un heap è merge-sort applicato all'heap stesso.

Esercizio 2.[11 punti]
Dato un albero binario, trovare un algoritmo in pseudocodice per calcolare, la sua larghezza. La larghezza di un albero è il numero massimo di nodi allo stesso livello o, equivalentemente, alla stessa distanza dalla radice. Calcolare il costo dell'algoritmo proposto e argomentare la sua correttezza.

Traccia della soluzione dell'esercizio 2.
Applichiamo una BFS sull'albero. La BFS memorizza su ogni nodo $v_i \in N$ la sua distanza $d(v_i)$ dalla radice e costa $\Theta(N + A)$ che su un albero è $\Theta(N)$ visto che $A = \Theta(N)$. Sia V un vettore che memorizza in posizione i il numero di nodi distanti i dalla radice dell'albero. Per costruire V basta scorrere tutti i nodi v_i e eseguire l'istruzione $V(d(v_i)) = V(d(v_i)) + 1$ (costo $\Theta(N)$). A questo punto basta calcolare il massimo di V (costo $\Theta(N)$). Costo totale $\Theta(N)$.

Esercizio 3.[11 punti]
Scrivere una funzione ricorsiva F_1 che prenda in input un vettore di n numeri e restituisca in output 0. F_1 deve avere costo computazionale $\Theta(n^2)$. Analizzare il costo computazionale della funzione proposta.
Scrivere una funzione ricorsiva F_2 che prenda in input un vettore di n numeri e restituisca in output 0. F_2 deve avere costo computazionale $\Theta(n \log(n))$. Analizzare il costo computazionale della funzione proposta.
Scrivere una funzione ricorsiva F_3 che prenda in input un vettore di n numeri e restituisca in output 0. F_3 deve avere costo computazionale $\Theta(1)$. Analizzare il costo computazionale della funzione proposta.

Utilizzare SOLAMENTE l'istruzione IF THEN ELSE, l'assegnamento e, eventualmente, la chiamata ad una funzione $G(n)$ che restituisce sempre il valore 1 e che ha costo computazionale $\Theta(n)$ (oltre che ovviamente il RETURN). Vietato utilizzare cicli (for, repeat, while , ...) o salti (goto, ...). Utilizzare la ricorsione.

Traccia della soluzione dell'esercizio 3.
Per semplicità assumeremo che le funzioni prendano in input la lunghezza del vettore.

$F_1(n)$
1 **if** $n \leq 10$ **return** 0
2 **else**
3 **return** $F_1(\lfloor n/2 \rfloor)+F_1(\lfloor n/2 \rfloor)+F_1(\lfloor n/2 \rfloor)+F_1(\lfloor n/2 \rfloor)$

Costo computazionale: $T(n) = 4\,T(\lfloor n/2 \rfloor) + c \Rightarrow T(n) = \Theta(n^2)$

$F_2(n)$
1 **if** $n \leq 10$ **return** 0
2 **else**
3 **return** $F_2(\lfloor n/2 \rfloor)+F_2(\lfloor n/2 \rfloor)+G(n) - 1$

Costo computazionale: $T(n) = 2\,T(\lfloor n/2 \rfloor) + n \Rightarrow T(n) = \Theta(n \log(n))$

$F_3(n)$
1 **if** $n \leq 10$ **return** 0
2 **else**
3 **return** $F_3(10)$

Costo computazionale: $T(n) = \Theta(1)$