Introduzione agli Algoritmi / Algoritmi I (A.A. 2009-2010)
Canali EO e PZ – Docenti: Calamoneri, Finocchi
Esame del 21 Luglio 2010

PARTE 1

Esercizio 1 (8 punti). Sia A un array (non ordinato) contenente n elementi. La frequenza $f(x)$ di un elemento x è il numero di occorrenze di x in A. Ad esempio, se $A = [5, 4, 1, 3, 1, 1, 3, 4, 5, 3, 4, 1]$, allora $f(1) = 4$, $f(3) = 3$, $f(4) = 3$ e $f(5) = 2$.

Progettare un algoritmo, il più possibile efficiente, per ordinare gli elementi nell'array A per frequenza crescente, facendo anche in modo che tutte le occorrenze di uno stesso elemento appaiano in posizioni consecutive. Ad esempio, se A è l'array mostrato sopra, una possibile soluzione potrebbe essere: $A = [5, 5, 3, 3, 3, 4, 4, 4, 1, 1, 1, 1]$. Discutere la correttezza e il tempo di esecuzione dell'algoritmo proposto.

Suggerimento: innanzitutto, progettate un algoritmo efficiente per calcolare le frequenze. Potete usare come subroutine algoritmi di ordinamento visti in aula.

Esercizio 2 (8 punti). Si consideri il seguente algoritmo `Partiz()` che, preso in input un array V di n valori, partiziona il sottoarray $V[i]...V[j]$, con $1 \leq i \leq j \leq n$, rispetto ad un valore p dato:

```
Partiz(array V, intero i, intero j, valore p) {
    if (i < j) {
        if (V[i] <= p) return Partiz(V, i + 1, j, p);
        else if (V[j] >= p) return Partiz(V, i, j - 1, p);
        else {
            scambia V[i] e V[j]
            return Partiz(V, i + 1, j - 1, p);
        }
     }
}
```

La chiamata iniziale è effettuata con indici $i = 1$ e $j = n$.

1. Valutare il tempo di esecuzione nel caso peggiore in funzione della dimensione n dell'array, mostrando il procedimento usato per la valutazione.
2. Il tempo di esecuzione (asintotico) nel caso migliore è lo stesso del caso peggiore? Motivare la risposta.

Introduzione agli Algoritmi / Algoritmi I (A.A. 2009-2010)
Canali EO e PZ – Docenti: Calamoneri, Finocchi
Esame del 21 Luglio 2010

PARTE 2

Esercizio 3 (8 punti). Sia H un maxHeap.

A) Se si inserisce in H prima un intero k e poi un altro intero $h \neq k$, si ottiene lo stesso albero che sia otterrebbe inserendo prima h e poi k? Cioé, l'inserimento in un maxHeap è commutativo? Dimostrare la risposta in positivo o fornire un controesempio.

B) Supporre che H, oltre alla classica proprietà di ordinamento a heap tra padre e figli, soddisfi anche la seguente proprietà di *ordinamento sui livelli*: se due nodi u e v sono sullo stesso livello e u si trova più a sinistra di v, allora $chiave(u) \leq chiave(v)$. Chiamiamo un maxHeap che gode di questa ulteriore proprietà un *maxHeap ordinato per livelli*. Considerare il seguente algoritmo per l'inserimento di una chiave k in un maxHeap ordinato per livelli:

1. se l'ultimo livello è pieno, inserisci k come nell'algoritmo di inserimento standard ed esci;
2. ricerca la posizione ordinata di k nell'ultimo livello (non pieno), mantenendo l'ordinamento sul livello;
3. sposta a destra di una posizione tutte le chiavi dell'ultimo livello maggiori di k;
4. inserisci k nella posizione vuota risultante e, se necessario, fai risalire k come nell'algoritmo di inserimento standard.

Solo una delle seguenti affermazioni è vera: quale?

(a) Il tempo di esecuzione dell'algoritmo è $O(\log n)$.
(b) L'algoritmo mantiene la proprietà di ordinamento a heap.
(c) L'algoritmo mantiene la proprietà di ordinamento sui livelli.

Confutare con un controesempio le due affermazioni false, e dimostrare la veridicità dell'affermazione rimanente.

Esercizio 4 (8 punti). Sia T un albero binario di ricerca di altezza h. Assumere che ogni nodo v di T contenga, oltre alla chiave ed ai puntatori ai figli, anche un campo *size* che memorizza il numero di nodi nel sottoalbero di T radicato in v.

1. Progettare un algoritmo efficiente che, ricevuta in input una chiave k, restituisca il numero di nodi di T con chiave $\leq k$.
2. Analizzare correttezza e tempo di esecuzione dell'algoritmo proposto.