Università degli Studi di Camerino – Laurea in Informatica
Prima Prova Parziale di **Algoritmi e Strutture Dati** - A -

Docente: Emanuela Merelli

16 dicembre 2010

Nome: Cognome: N.Matricola:

1. (3 points) Dimostrare che $f(n) = \Theta(k(n))$ e $h(n) = \Theta(k(n))$ allora $f(n) = \Theta(h(n))$.

   Sol.: cfr. Esercizio 4.6 della raccolta "Esercizi utili"

2. (3 points) Dimostrare che

   $$\sum_{i=1}^{n} \log n = O(n \log n)$$

   Sol.:
   $$\sum_{i=1}^{n} \log n = \log n + \log n + \log n ... + \log n = n \log n$$

3. (3 points) Si risolva la seguente ricorrenza con il teorema Master:
   $$T(n) = 3T(\tfrac{n}{2}) + 4n^2\sqrt{n}$$
   Sol.: cfr. Esercizio 4.14 della raccolta "Esercizi utili"

4. (3 points) Dimostrare, utilizzando il principio di induzione, che

   $$\sum_{i=1}^{n+1}(i-1)^2 = \tfrac{n(n+1)(2n+1)}{6}$$

   Sol.: cfr. Esercizio A.4 della raccolta "Esercizi sull'induzione"

5. (3 points) Rispondere alle seguenti domande:

   - Qual e' la complessita' dell'algoritmo di ricerca sequenziale in funzione del numero n di elementi?
   - Qual e' la complessita' nel caso peggiore dell'algoritmo di ricerca binaria in funzione del numero n di elementi?
   - Quanti confronti esegue l'algoritmo di ricerca binaria nel caso migliore?
   - Quanti confronti esegue l'algoritmo di ricerca binaria nel caso medio?

   Sol.: 4.1: O(n); 4.2: O(lg n); 4.3: O(1); 4.4: O(log n)

6. (5 points) Mettere in ordine di velocità di crescita le seguenti funzioni in n e determinare il valore di n_0 per il quale l'ordinamento proposto è valido per ogni $n \geq n_0$:

   $$\left(\; n!, \quad n^n, \quad \log n^2, \quad n^3, \quad \log 2n, \quad 2n, \quad 2^n, \quad \log n^n \; \right)$$

   Sol:

   $$\left(\; \log 2n, \quad \log n^2, \quad 2n, \quad \log n^n, \quad n^3, \quad 2^n, \quad n!, \quad n^n \; \right)$$

7. (3 points) Qual e' il tempo richiesto dall'algoritmo di visita in profondita' di un albero binario in funzione del numero n di nodi?

   Sol: $\Theta(n)$

8. (3 points) Sia V un array di dimensione m contenente m valori interi e z un valore intero. Scrivere un algoritmo, la cui complessita' nel caso peggiore e' $O(m \log m)$, che restituisca vero solo se esistono due elementi in V la cui differenza e' z.

   Sol.: cfr. Esercizio 4.36 della raccolta "Esercizi utili"

9. (3 points) Si considerino un albero binario di ricerca T ed un array ordinato A ciascuno di n elementi. Proporre un algoritmo che produca in output un array ordinato B che contenga gli elementi di T e quelli di A.

   Sol.: cfr. Esercizio 4.36 della raccolta "Esercizi utili"

10. (3 points) Eseguire il MergeSort sull'input 7, 6, 5, 3, 2, 0, 8, 9, 4, 1. Fornire una rappresentazione grafica.

    Sol: cfr. lezione sugli algoritmi di ordinamento

11. (3 points) Mostrare che, scegliendo opportunamente le priorita' degli elementi, e' possibile fare in modo che una coda con priorita' si comporti come una coda o come una pila

    Sol: cfr. commenti alla lezione su Heap

12. (3 points) Sia dato un array A di n interi. Progettare un algoritmo divide et impera che sia in grado di restituire il massimo ed il minimo in A in un tempo $O(n)$.

    Sol:
    Sia *time* un contatore che viene incrementato ad ogni operazione. Scegliendo come priorita' degli elementi il valore *time* al tempo del loro inserimento, una coda con priorita' si comportera' come una coda. Infatti, l'elemento con priorita' minima sara' sempre quello inserito piu' indietro nel tempo. In questo primo scenario, l'operazione *insert* sara' equivalente all'operazione *enqueue* e l'operazione $findMin$ sara' equivalente all'operazione *dequeue*. Scegliendo invece come priorita' degli elementi il valore $-time$ (oppure $1/time$) al tempo del loro inserimento, una coda con priorita' si comportera' come una pila. Infatti, l'elemento con priorita' minima sara' sempre quello inserito piu' recentemente. In questo secondo scenario, l'operazione *insert* sara' equivalente all'operazione *push* e l'operazione $findMin$ sara' equivalente all'operazione *pop*.

13. (3 points) Considerare la seguente sequenza di numeri: 10 30 200 320 800 220 150 400 600 930 810 420 assumendo che siano memorizzati in un array secondo la rappresentazione posizionale, stabilire, motivando la risposta, se la sequenza rappresenta un MinHeap. Nel caso non risultasse tale, indicare qual è il sottoalbero che non è un MinHeap. Descrivere le operazioni da effettuare per costruire l'equivalente MinHeap. Stabilire l'ordine della complessità.

    Sol: cfr. lezione su Heap

14. (3 points) Si consideri una tabella hash di dimensione $m = 10$ inizialmente vuota. Si mostri il contenuto della tabella dopo aver inserito la seguente sequenza di valori 44, 32, 26, 49, 36, 74, 77, 27, 5, 55. Si assuma che le collisioni vengano gestite mediante indirizzamento aperto utilizzando come funzione hash $h(k, i) = (h'(k) + 3i + i^2) \bmod m$ dove la funzione hash ordinario $h'(k) = k \bmod m$.

    Sol: cfr. lezione su Hash Table

Question:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total
Points:	3	3	3	3	3	5	3	3	3	3	3	3	3	3	44
Score:															