Esercizi per il corso di Algoritmi, anno accademico 2011/12

Esercizi sulla Tecnica Divide et Impera

N.B. Tutti gli algoritmi vanno scritti in pseudocodice (non in Java, nè in C++, etc.). Di tutti gli algoritmi và presentata una discussione, anche informale, del loro funzionamento e del perchè calcolano l'output richiesto. Inoltre, di tutti gli algoritmi và analizzata la complessità di tempo, giustificando le affermazioni fatte. Infine, gli algoritmi vanno necessariamente scritti usando la tecnica richiesta.

1. *Esercizio:* Dato un vettore di interi $A[0 \dots n-1]$, progettare un algoritmo basato sulla tecniva Divide et Impera per calcolare la quantità

$$S = A[0]2^0 + A[1]2^1 + A[2]2^2 + \cdots A[n-1]2^{n-1}.$$

$\diamond$

2. *Esercizio:* Dato un vettore di interi $A[1 \dots n]$ ed un intero k, progettare un algoritmo basato sulla tecnica Divide et Impera per calcolare la quantità

$$(A[1] * A[2] + A[2] * A[3] + A[3] * A[4] \cdots + A[n-2] * A[n-1] + A[n-1] * A[n]) \bmod k.$$

$\diamond$

3. *Esercizio:* Dato un vettore di interi $A[1 \dots n]$, progettare un algoritmo basato sulla tecnica Divide et Impera per calcolare la quantità

$$S = (A[1] + n) + (A[2] + 2n) + (A[3] + 3n) + \cdots (A[n] + n^2).$$

$\diamond$

4. *Esercizio:* Sia A un vettore di n interi. Si dice che A è continuo se per ogni $i = 1, 2, \dots, n-1$, vale che $|A[i+1] - A[i]| \leq 1$. Si dice zero del vettore un indice k tale che $A[k] = 0$.
Dato un vettore A di $n \geq 2$ interi continuo tale che $A[1] \leq 0$ e $A[n] > 0$, provare che A ha almeno uno zero.
Progettare un algoritmo basato sulla tecnica Divide et Impera che, dato un vettore A di $n \geq 2$ interi continuo e tale che $A[1] < 0$ e $A[n] > 0$, trovi uno zero in tempo $O(\log n)$.

$\diamond$

5. *Esercizio:* Dato un vettore $A[1...n]$, con $A[i] \in \{0, 1\}$, per ogni $i = 1, \dots, n$, progettare ed analizzare un algoritmo basato sulla tecnica Divide et Impera che restituisca VERO se esiste $i \in \{1, \dots, n-1\}$ tale che $A[i] = 1$ e $A[i+1] = 0$, FALSO altrimenti.

$\diamond$

6. *Esercizio:* Dato un vettore $A[1...n]$, con $A[i] \in \{0,1,2\}$, per ogni $i = 1,\ldots,n$, progettare ed analizzare un algoritmo basato sulla tecnica Divide et Impera che restituisca VERO se esiste $i \in \{1,\ldots,n-2\}$ tale che $A[i] = 2$ e $A[i+1] = 0, A[i+2] = 1$, FALSO altrimenti.

$\diamond$

7. *Esercizio:* Dato un vettore $A[1...n]$, progettare ed analizzare un algoritmo basato sulla tecnica Divide et Impera che restituisca la quantità $3A[1] + A[2] + 3A[3] + \ldots + (2 + (-1)^{n+1}A[n])$

$\diamond$

8. *Esercizio:* Dati due alberi binari T ed S, si scriva un algoritmo basato sulla tecnica Divide et Impera che determini se i due alberi sono uguali o meno. L'input all'algoritmo è costituito dai due puntatori alla radice di T ed S, rispettivamente. [Chiarimento: per alberi "uguali" si intendono alberi con la stessa struttura, ovvero alberi che se disegnati apparirebbero identici].

$\diamond$

9. *Esercizio:* Dato un albero binario con puntatore alla radice z, per ogni nodo v dell'albero si definisca il suo peso $p(v)$ come

$$p(v) = \begin{cases} 1, & \text{se } v \text{ è una foglia} \\ 1 + \text{prodotto dei pesi dei suoi figli}, & \text{altrimenti} \end{cases}$$

Si progetti ed analizzi un algoritmo basato sulla tecnica Divide et Impera che calcoli per ogni nodo v dell'albero il suo peso $p(v)$.

$\diamond$

10. *Esercizio:* Si consideri la seguente variante dell'algoritmo di ricerca binaria in un array ordinato A. Ad ogni passo l'array viene diviso in tre parti e l'elemento la cui posizione deve essere determinata in A viene confrontato con due elementi. Si scriva con precisione il relativo algoritmo, analizzandone la complessità di tempo.

$\diamond$

11. *Esercizio:* Sia dato un albero binario T, con puntatore alla radize z. Ad ogni nodo x dell'albero è associato un campo numerico $key[x]$. Si scriva un algoritmo basato sulla tecnica Divide et Impera che, preso in input un puntatore alla radice di T ed un numero k, restituisca in output il numero di nodi x in T che hanno $key[x] = k$.

$\diamond$

12. *Esercizio:* Sia dato un albero binario T con puntatore alla radice z. Si scriva un algoritmo basato sulla tecnica Divide et Impera che conti il numero di foglie di T.

$\diamond$

13. *Esercizio:* Sia dato un albero binario T, con puntatore alla radize z. Ad ogni nodo x dell'albero è associato un campo numerico $key[x]$. Si scriva un algoritmo basato sulla tecnica Divide et Impera che, preso in input un puntatore alla radice di T ed un numero k, restituisca in output tutti i nodi x in T che hanno $key[x] \geq k$.

◇

14. *Esercizio:* Sia dato un albero binario T con puntatore alla radice z. Si scriva un algoritmo basato sulla tecnica Divide et Impera che calcoli la quantità

$$\sum_{x:x \text{ è una foglia di } T} d_T(x),$$

dove $d_T(x)$ è la distanza (ovvero il livello) della foglia x dalla radice di T.

◇

15. *Esercizio:* Sia dato un albero binario T con puntatore alla radice z. Ad ogni nodo x dell'albero è associato un campo numerico $key[x]$. Si scriva un algoritmo basato sulla tecnica Divide et Impera che calcoli la quantità

$$\sum_{x:x \text{ è una foglia di } T} key[x].$$

◇

16. *Esercizio:* Sia dato un albero binario T con puntatore alla radice z. Ad ogni nodo x dell'albero è associato un campo numerico $key[x]$. Si scriva un algoritmo basato sulla tecnica Divide et Impera che calcoli la quantità

$$\sum_{x:x \text{ è una foglia di } T} key[x] * d_T(x).$$

dove $d_T(x)$ è la distanza (ovvero il livello) della foglia x dalla radice di T.

◇

17. *Esercizio:* Sia dato un albero binario T con puntatore alla radice z. Per ogni nodo x dell'albero vale che $key[x] \in \{0, 1\}$. Si scriva un algoritmo basato sulla tecnica Divide et Impera che calcoli lo XOR (ovvero l'OR esclusivo) dei valori $key[x]$.

◇

18. *Esercizio:* Si progetti e si analizzi un algoritmo basato sulla tecniva Divide et Impera che, preso in input un vettore di interi $A[1 \ldots n]$, restituisca in output il valore $\min_{i,j} |A[i] - A[j]|$.

◇

19. *Esercizio:* Si consideri il seguente problema: la Ditta ACME è stata quotata in borsa, ed il valore delle sue azioni sono state tabulate per tutto l'anno. Sia A il valore di una azione di ACME al primo Gennaio, e sia B il corrispondente valore al 31 Dicembre.

    (a) Se $A > B$, argomentare che c'è stato un giorno dell'anno in cui l'azione di ACME è stata quotata ad un valore inferiore al giorno precedente;

(b) formalizzando opportunamente il problema in termini algoritmici (cioè definendo con precisione chi sono gli input e gli output al problema), sia dia un algoritmo di complessità logaritmica nella taglia dell'input che, sotto l'ipotesi che $A > B$, determini un giorno dell'anno in cui l'azione di ACME è stata quotata ad un valore inferiore al giorno precedente.

$\diamond$

20. *Esercizio:* Sia $A[1 \ldots n]$ un vettore ordinato che é stato shiftato k posizioni a sinistra. Ad esempio, il vettore $[15, 18, 28, 30, 35, 42, 1, 7]$ é un vettore ordinato che é stato shiftato $k = 2$ posizioni a sinistra, mentre il vettore $[30, 35, 42, 1, 7, 15, 18, 28]$ é un vettore ordinato che é stato shiftato $k = 5$ posizioni a sinistra.

    (a) Supponendo di avere A e k in input, dare un algoritmo che determina il minimo in A in tempo $O(1)$

    (b) Supponendo di avere *solo* il vettore A in input, dare un algoritmo che determina il minimo in A in tempo $O(\log n)$

$\diamond$

21. *Esercizio:* Un vettore di interi distinti $A[1...n]$ é detto *unimodulare* se esiste un indice i per cui $A[1] < A[2] < \ldots < A[i-1] < A[i]$ e $A[i] > A[i+1] > \ldots > A[n]$. Progettare un algoritmo che determini, in tempo $O(\log n)$, il valore massimo di un vettore unimodulare.

$\diamond$

22. *Esercizio:* Dato un vettore *ordinato* di interi $A[1 \ldots n]$ ed un intero N, si progetti e si analizzi un algoritmo basato sulla tecnica Divide et Impera che, preso in input il vettore A ed il numero N determini se esistono o meno due indici i e j per cui $A[i] \times A[j] = N$.

$\diamond$

23. *Esercizio:* Si descriva e si analizzi l'algoritmo di complessità $O(n^{\log_2 3})$ per la moltiplicazione di due numeri di n bits.

$\diamond$

24. *Esercizio:* Sia data una matrice $n \times n$ di interi in cui gli elementi di ogni riga sono ordinati in senso crescente, e anche gli elementi di ogni colonna sono ordinati in senso crescente. Si progetti un algoritmo per determinare se un dato intero k é presente nella matrice o meno, e se ne analizzi la complessitá.

$\diamond$

25. *Esercizio:* Data una matrice $n \times n$ di interi, con n potenza di 2, si scriva un algoritmo che determina il minimo nella matrice, usando la tecnica "Divide et Impera". Se ne analizzi la complessitá.

$\diamond$

26. *Esercizio:* Dato un vettore A di n interi, si dice *salto* in A un indice $i, 1 \leq i < n$, tale che $A[i+1] - A[i] \geq 2$. Provare che ogni vettore A di $n \geq 2$ interi per cui $A[n] - A[1] \geq n$ possiede almeno un salto. Progettare un algoritmo che, dato un vettore A di $n \geq 2$ interi per cui $A[n] - A[1] \geq n$, trovi un salto in $O(\log n)$ tempo.

Esercizi sull' Ordinamento e Selezione

1. *Esercizio*: Si esegua l'algoritmo SELECT$(A, 4)$ sull'array $A = [12, 3, 7, 2, 14, 9, 15, 5, 21, 6, 1, 10, 8, 4]$ riportando chiaramente, per ogni passo dell'algoritmo, le partizioni dell'array. Si assuma che ad ogni iterazione, l'algoritmo SELECT scelga come pivot l'elemento più a sinistra del sottoarray in quel momento in considerazione.

◇

2. *Esercizio*: Si esegua l'algoritmo QUICKSORT sull'array $[24, 33, 25, 45, 11, 12, 23, 13]$, riportando chiaramente, per ogni passo dell'algoritmo, le partizioni dell'array. Si assuma che ad ogni iterazione, l'algoritmo QUICKSORT scelga come pivot l'elemento più a sinistra del sottoarray in quel momento in considerazione.

◇

3. *Esercizio*: Sia $A[1 \dots n]$ un array di interi. Si dia un algoritmo che riordini gli elementi di A in modo tale che tutti gli elementi negativi appaiano alla sinistra di tutti gli elementi positivi. L'algoritmo deve avere complessità $\Theta(n)$ nel caso peggiore e **non** deve usare array ausiliari.

◇

4. *Esercizio*: Si supponga di disporre di un "super calcolatore" che sia capace di effettuare il merge di due sequenze ordinate, ciascuna lunga n, in tempo $\sqrt{n}$. Si scriva un algoritmo ricorsivo che usa questo super-calcolatore per ordinare un vettore lungo n. Si scriva una relazione di ricorrenza per descrivere il tempo di esecuzione di tale algoritmo, si risolva la equazione di ricorrenza usando i teoremi generali per la risoluzione di equazioni di ricorrenza visti a lezione.

◇

5. *Esercizio*: Si supponga di disporre di un "super calcolatore" che sia capace di calcolare il massimo di 3 elementi in un solo passo. Sia dia un algoritmo efficiente per il calcolo dell'elemento massimo in un vettore $A[1 \dots n]$, nell'ipotesi di utilizzare tale super calcolatore. L'analisi della complessitá dell'algoritmo dovrebbe essere quanto piú precisa possibile, e non solo di tipo asintotico.

◇

6. *Esercizio*: Sia $A[1, \dots, n]$ un vettore contenente $n = 3m$ interi, tutti distinti tra di loro. Si consideri il problema di determinare gli elementi di A maggiori o uguali ad almeno $n/4$ interi in A e minori o uguali ad almeno $n/4$ interi in A.
   - Si proponga un algoritmo lineare per risolvere il problema proposto;
   - si discuta la correttezza e la complessità dell' algoritmo definito.

◇

7. *Esercizio*: Sia $A[1 \dots n]$ un array tale che i primi $n - \sqrt{n}$ elementi siano giá ordinati. Si scriva un algoritmo che ordini l'intero array A in tempo sostanzialmente inferiore a $n \log n$.

◇

8. *Esercizio*: Sia $A[1 \dots n]$ un vettore tale che $\forall i$ vale che $A[i] \in \{0, 1, 2\}$ Scrivere un algoritmo che ordina A in tempo $O(n)$

◇

9. *Esercizio*: Sia $A[1 \dots n]$ un array di interi distinti, con $n = k \times a$. Si dia un algoritmo efficiente per suddividere l'array A in k sottoarray $A_1, A_2, \dots, A_k$, ciascuno composto di a elementi, tale che se $i < j$ allora ogni elemento nell'array A_i é minore di ogni elemento nell'array A_j. Gli elementi all'interno di ogni sottoarray non devono essere necessariamente essere ordinati. (Sugg.: si possono usare all'interno dell'algoritmo chiamate all'algoritmo SELECT).

◇

10. *Esercizio*: Sia dato un vettore di interi $A[1 \dots 2n]$. Sia dia un algoritmo di complessitá $O(n \log n)$ che determini una coppia di elementi x, y, con $x \leq y$, di A per cui il valore $y - x$ é massimo. Si generalizzi l'esercizio al caso in cui occorre suddividere gli elementi di A tra due vettori $B[1 \dots n]$ e $C[1 \dots n]$ tale che la differenza

$$\sum_{i=1}^{n} C[i] - \sum_{i=1}^{n} B[i]$$

sia massima possibile. Si giustifichi la risposta.

◇

11. *Esercizio*: Si consideri la seguente variante di Quicksort. Si prendano due elementi a, b dell'array $A[1 \dots n]$ da ordinare, e si partizioni A in tre sottoarray A_1, A_2, e A_3, dove $A_1 = \{x \in A \ : \ x < a \ \text{ e } x < b\}$, $A_3 = \{x \in A \ : \ x > a \ \text{ e } x > b\}$, A_3 contiene i restanti elementi.

    (a) Si scriva lo pseudocodice per questo algoritmo

    (b) Quanti confronti usa l'algoritmo per partizionare l'array A in A_1, A_2, e A_3?

    (c) Si assuma che l'algoritmo di partizione usato ritorni sempre una partizione per cui $|A_1| = |A_2| = |A_3| = n/3$. Scrivere una relazione di ricorrenza per il numero di confronti $T(n)$ effettuati dall'algoritmo di ordinamento tipo Quicksort cosí ottenuto.

    (d) Si trovi una costante a per cui valga $T(n) \leq an \log n$ (per quest'ultimo punto si suggerisce di procedere per induzione).

◇

12. *Esercizio*: Si consideri la seguente variante di MERGESORT. Dato un array $A[1 \dots n]$ si divida A in tre sottovettori di eguale grandezza, si ordini ciascun sottovettore, indi si effettui il merge dei primi due sottovettori ordinati in un unico sottovettore, e successivamente si effettui il merge del sottovettore cosí ottenuto con il terzo sottovettore. Si assuma di disporre di una procedura MERGE che effettua il merge di due vettori ordinati di lunghezza n_1 e n_2 in tempo $n_1 + n_2$. Si assuma anche per semplicitá che n sia potenza di tre.

    (a) Si scriva lo pseudocodice dell'algoritmo sopra esposto in maniera informale;

    (b) Si analizzi la complessitá dell'algoritmo.

◇

13. *Esercizio* Dato un vettore $A[1...n]$, si descriva l'algoritmo randomizzato $\textsc{Select}(A, k)$ per il calcolo dell'elemento di rango k di A e se ne analizzi la complessità.

◇

14. *Esercizio* Si descriva l'algoritmo randomizzato QuickSort e se ne analizzi la complessità.

◇

15. *Esercizio* Si descriva un algoritmo che, prendendo in input un vettore di interi distinti $A[1 \dots n]$ ed un intero $k \leq n$, produce in output i k elementi più grandi di $A[1 \dots n]$, dal più grande al più piccolo. Il tutto in tempo $O(n + k \log k)$ nel caso peggiore (**N.B.**: l'algoritmo da progettare può effettuare chiamate al suo interno ad algoritmo visti a lezione).