

Esercizi

Esercizio 1

Progettare un algoritmo per ordinare **in loco** un array a di n interi, il cui valore può essere solo 0, 1 o 2. L'algoritmo deve richiedere tempo lineare nel caso pessimo e può solo scambiare elementi. In particolare, non può usare contatori per mantenere il numero di elementi di un certo valore.

Esercizio 2

Progettare un algoritmo efficiente di tipo divide et impera per determinare se un array non ordinato di n interi, contenente solo 0 e 1, contiene più 0 di 1. Analizzare la complessità dell'algoritmo trovato, e dimostrarne la correttezza.

Esercizio 3

Si consideri la seguente relazione di ricorrenza

$$G(n) = \begin{cases} 1 & \text{per } n = 0 \\ 2 & \text{per } n = 1 \\ G(n-1) + 2G(n-2) & \text{per } n > 1 \end{cases}$$

- Trovare i valori di $G(n)$, per $n = 2, 3, 4, 5$.
- Dal punto precedente, intuire la soluzione esatta di $G(n)$ e provarla per induzione.
- Dato un intero n , descrivere un algoritmo efficiente per il calcolo di $G(n)$ analizzandone la complessità in tempo.

Esercizio 4

Progettare un algoritmo di ordinamento che si comporti come MergeSort, ma che divida l'array ricorsivamente in tre parti anziché in due. Scrivere lo pseudocodice della nuova procedura *MergeSort3*. Descrivere a parole la nuova procedura *Fusione3*. Impostare e risolvere l'equazione di ricorrenza associata.