

# Esercizi

## Esercizio 1

Progettare un algoritmo di costo in tempo  $O(\log n)$  per il calcolo dell' $n$ -simo numero di Fibonacci  $F_n$ , basato sul seguente lemma:

**Lemma**

$$\text{Sia } A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}. \text{ Allora } A^{n-1} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} = \begin{pmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{pmatrix}.$$

## Esercizio 2

Progettare un algoritmo efficiente per verificare se in un array  $a$  di  $n$  interi positivi esistono  $i$  e  $j$  tali che  $a[j] = 2 * a[i]$ , e analizzarne la complessità.

## Esercizio 3

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.

## Esercizio 4

Progettare un algoritmo efficiente, di tipo divide et impera, per verificare se un array di caratteri contiene la sequenza 'a' 'b' memorizzata in posizioni adiacenti, e analizzarne la complessità.

## Esercizio 5

Sia  $a$  un array di  $n$  interi distinti, tale che esiste una posizione  $j$ ,  $0 \leq j < n$ , per cui:

- gli elementi nel segmento  $a[0, j]$  sono in ordine decrescente;
- gli elementi in  $a[j+1, n-1]$  sono in ordine crescente;
- $a[j] < a[j+1]$ , se  $j < n - 1$ .

Si consideri il problema di trovare la posizione  $j$ .

1. Dimostrare che, un qualunque algoritmo che risolve il problema suddetto mediante confronti, richiede tempo  $\Omega(\log n)$  al caso pessimo.
2. Descrivere un algoritmo **ottimo** di tipo *divide-et-impera* per il problema precedente. Calcolare la complessità al caso pessimo dell'algoritmo indicando, e risolvendo, la corrispondente relazione di ricorrenza.

## Esercizio 6

Scrivere un algoritmo divide et impera che, dato un array  $a$  **ordinato** di interi **distinti** (anche negativi), verifichi se esiste un indice  $i$  tale che  $a[i] = i$ . Analizzare la complessità dell'algoritmo proposto.

## Esercizio 7

Dato l'insieme  $A = \{2, 6, 4, 3, 5\}$  di somma  $2s = 20$ , simulare l'algoritmo di programmazione dinamica **Partizione**, mostrando il contenuto della tabella che l'algoritmo riempie dinamicamente e la soluzione trovata dall'algoritmo.

## Esercizio 8

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.