Esercizi

Esercizio 1
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 2
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 3
Progettare un algoritmo di tipo *divide-et-impera* per calcolare a^n con $O(\log n)$ moltiplicazioni.
[*Suggerimento:* $a^n = (a^{n/2})^2$ per n pari, e $a^n = a\,(a^{n/2})^2$ per n dispari.]

Esercizio 4
Progettare un algoritmo di tipo *divide-et-impera* per contare il numero di elementi pari in un array di interi, e analizzarne la complessità.

Esercizio 5
Dato un array di n interi positivi, progettare un algoritmo efficiente per verificare se esistono due elementi nell'array la cui somma è k.

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

1. Descrivere un algoritmo che, ricevuto in ingresso a, trova la posizione j in tempo lineare.
2. Dimostrare che, un qualunque algoritmo che risolve il problema suddetto mediante confronti, richiede tempo $\Omega(\log n)$ al caso pessimo.
3. 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 8
Un array a di n interi distinti si definisce **ciclicamente ordinato** se esiste un *indice* i, $0 \leq i < n$, tale che la sequenza $a[i], a[i+1], \ldots, a[n-1], a[0], \ldots, a[i-1]$ è ordinata in modo crescente. Ad esempio, l'array $a = [12, 14, 20, 1, 3, 7, 10, 11]$ è ciclicamente ordinato per $i = 3$. Si consideri il problema di trovare la posizione i.

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