Esercizi Divide-et-impera

1. **Punto fisso**: Scrivere un algoritmo che prenda in input un vettore ordinato A contenente n interi distinti e restituisca **true** se e solo se esiste un indice i tale che $A[i] = i$, in tempo $O(\log n)$.
2. **Chi manca?**: Scrivere un algoritmo che prenda in input un vettore ordinato $A[1 \dots n]$ contenente n elementi interi distinti appartenenti all'intervallo $1 \dots n+1$ e restituisca in tempo $O(\log n)$ l'unico intero dell'intervallo $1 \dots n+1$ che non compare in A.
3. **Vettori uni-modulari**: Un vettore di interi distinti A è detto unimodulare se esiste un indice h tale che $A[1] > A[2] > \dots > A[h-1] > A[h]$ e $A[h] < A[h+1] < A[h+2] < \dots < A[n]$, dove n è la dimensione del vettore. Scrivere un algoritmo che prenda in input un vettore unimodulare e restituisca il valore minimo del vettore in tempo $O(\log n)$.

Discutere correttezza e complessità degli algoritmi proposti.

Samarcanda

Nel gioco di Samarcanda, ogni giocatore è figlio di una nobile famiglia della Serenissima, il cui compito è di partire da Venezia con una certa dotazione di denari, arrivare nelle ricche città orientali, acquistare le merci preziose al prezzo più conveniente e tornare alla propria città per rivenderle.

Scrivere un algoritmo che prenda in input un vettore P contenente n interi in cui $P[i]$ è il prezzo di una certa merce al giorno i e restituisca il guadagno massimo $P[y] - P[x]$ che si può ottenere comprando la merce nel giorno x e rivenendola il giorno y, con $x < y$.

Discutere correttezza e complessità dell'algoritmo proposto.

Per fare un albero (binario di ricerca) ci vuole...

Dato un vettore V contenente n interi ordinati e distinti, scrivere un algoritmo che restituisca un albero binario di ricerca di altezza minima.

Discutere correttezza e complessità dell'algoritmo proposto.

Spoiler alert!

Esercizi Divide-et-Impera

Una spiegazione più approfondita del funzionamento di missing(), fixedPoint(), vum() si trova nel file `12 - Divide-et-impera` che si trova a questo indirizzo:
`http://cricca.disi.unitn.it/montresor/teaching/asd/materiale/esercizi/esercizi-argomento/`
Tutte le soluzioni si basano su ricerca dicotomica, quindi hanno complessità pari a $O(\log n)$.

Punto fisso

```
int fixedPoint(int[] A, int n)
return fixedPointRec(A, 1, n)
```

```
boolean fixedPointRec(int[] A, int i, int j)
if i > j then
    return false
int m = ⌊(i + j)/2⌋
if A[m] == m then
    return true
else if A[m] < m then
    return fixedPointRec(A, m + 1, j)
else
    return fixedPointRec(A, i, m − 1)
```

Chi manca?

```
int missing(int[] A, int n)
if A[n] == n then
    return n + 1
else
    return missingRec(A, 1, n)
```

```
int missingRec(int[] A, int i, int j)
if i == j then
    return i
int m = ⌊(i + j)/2⌋
if A[m] == m then
    return missingRec(A, m + 1, j)
else
    return missingRec(A, i, m)
```

Vettori unimodulari

```
int vum(int[] A, int n)
return vumRec(A, 1, n)
```

```
int vumRec(int[] A, int i, int j)
if i == j then
    return A[i]
int m = ⌊(i + j)/2⌋
if A[m] < A[m + 1] then
    return vumRec(A, i, m)
else
    return vumRec(A, m + 1, j)
```

Samarcanda

Struttura della soluzione:

- Si divide il vettore a metà
- Si applica ricorsivamente la soluzione nelle due metà, ottenendo la migliore soluzione nella metà destra e nella metà sinistra
- Da questo approccio, non vengono considerate le soluzioni in cui si acquista nella metà sinistra del vettore, si vende nella metà destra
- Si cerca quindi il minimo a sinistra e il massimo a destra e si applica ricorsivamente la soluzione
- Caso base: un elemento solo, che ovviamente dà origine a zero come massimo guadagno

Samarcanda

samarcanda(**int**[ ] A, **int** i, **int** j)

if $i \geq j$ **then**
  return 0
$m = \lfloor (i + j)/2 \rfloor$
int $maxLeft = \text{samarcanda}(L, i, m)$
int $maxRight = \text{samarcanda}(L, m + 1, j)$
int $ml = \min(A, i, m)$
int $mr = \max(A, m + 1, j)$
int $maxCross = \max(0, mr - ml)$
return $\max(maxLeft, maxRight, maxCross)$

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/2) + n & n > 1 \end{cases}$$
$$= \Theta(n \log n)$$

Samarcanda

- La soluzione così proposta non è quella ottima, perché il problema può essere risolto in tempo $\Theta(n)$
- Un possibile approccio consiste nel calcolare il vettore delle differenze fra le quotazioni, e applicare la somma massimale vista alla prima lezione. Costo: $\Theta(n)$.
- Altrimenti, è facile progettare una soluzione ad-hoc per il problema

Per fare un albero (binario di ricerca) ci vuole...

```
TREE build-tree-rec(int[] A, int i, int j)
if i ≤ j then
    int m = ⌊(i + j)/2⌋
    TREE T = new TREE
    T.left = build-tree-rec(A, i, m − 1)
    T.right = build-tree-rec(A, m + 1, j)
    T.key = V[m]
    return T
else
    return nil
```

L'equazione di ricorrenza è pari a: $T(n) = 2T(n/2) + 1$, che dà origine a $T(n) = \Theta(n)$.

Per fare un albero (binario di ricerca) ci vuole...

Correttezza: per induzione sulla dimensione dell'albero e per il fatto che qualunque altra divisione dei vali fra sottoalbero sinistro e sottoalbero destro dà origine a sottoalberi di altezza maggiore o uguale.