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 $A[1 \ldots n]$ contenente n elementi interi distinti appartenenti all'intervallo $1 \ldots n+1$ e restituisca in tempo $O(\log n)$ l'unico intero dell'intervallo $1 \ldots n+1$ 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 $A[1] > A[2] > \ldots > A[h-1] > A[h]$ e $A[h] < A[h+1] < A[h+2] < \ldots < A[n]$, 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.

Vicini-vicini – difficile

Siano dati due vettori $x[1 \ldots n]$ e $y[1 \ldots n]$ che rappresentano le coordinate cartesiane di n punti nel piano; ovvero il punto i-esimo è in posizione $(x[i], y[i])$. Scrivere un algoritmo che restituisca, fra tutte le coppie, quella con minima distanza.

Note:

- Esiste ovviamente una soluzione banale $O(n^2)$
- Esiste un algoritmo di costo $O(n \log n)$ basato su divide-et-impera, per niente facile da trovare
- Esiste un algoritmo probalistico di costo $O(n)$
- Questo problema ha applicazioni nei software GIS