Prof. L. Margara **Scritto del 15/9/2023**

Algoritmi e Strutture Dati

Esercizio 1.[11 punti]
Data una sequenza di n numeri interi $x_1, ..., x_n$ diciamo che (x_i, x_{i+2}) sono una coppia di numeri associati se $x_i + x_{i+2} = x_{i+1}$. Ad esempio nella sequenza $(3, -4, 8, 9, 1, -8, -31, -23)$ ci sono 3 coppie di numeri associati: $(8, 1)$, $(9, -8)$ e $(-8, -23)$. Scrivere un algoritmo **ricorsivo** che utilizzi la **tecnica divide-et-impera** (nella quale il vettore in input si spezza ricorsivamente in parti uguali) e che calcoli quante coppie di numeri associati sono contenute in una sequenza di n numeri interi $x_1, ..., x_n$ ricevuta in input. Vietato usare While, For, Repeat, Goto e qualunque altro costrutto sintattico che permetta di implementare cicli. Valutare in ordine di grandezza il costo computazionale dell'algoritmo proposto impostando e risolvendo l'equazione ricorsiva associata.

Traccia Soluzione Esercizio 1.

ContaCoppie(A,i,j)
    if $j - i \leq 1$ **then return** 0
    if $j - i = 2$ **then if** $A[i] + A[i+2] = A[i+1]$ **then return** 1
                                              else return 0
    $k = \lfloor (i+j)/2 \rfloor$
    // se arrivo qui, $A[i..k]$ e $A[k+1..j]$ contengono entrambi almeno 2 elementi
    $c = \text{ContaCoppie}(A, i, k) + \text{ContaCoppie}(A, k+1, j)$
    if $A[k-1] + A[k+1] = A[k]$ **then** $c = c + 1$
    if $A[k] + A[k+2] = A[k+1]$ **then** $c = c + 1$
    return c

Equazione Ricorsiva:

$$T(n) = \begin{cases} c_1 & \text{if } n \leq 3, \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + c_2 & \text{if } n > 3. \end{cases}$$

Soluzione: $T(n) = \Theta(n)$

Esercizio 2.[11 punti]
Descrivere un algoritmo che ordina n numeri compresi tra 0 e $n^5 - 1$ in tempo $O(n)$. Spiegare bene il procedimento e motivare tutti i passaggi.

Traccia Soluzione Esercizio 2.

Un numero compreso tra 0 e $n^5 - 1$ ha al massimo 5 volte le cifre di un numero compreso tra 0 e $n - 1$. Ogni numero compreso tra 0 e $n^5 - 1$ puó quindi essere visto come un numero di 5 cifre in base n. A questo punto applichiamo il radix sort sulle 5 cifre in base n. Su ogni singola colonna di cifre applichiamo il counting sort (le cifre sono comprese tra 0 e $n - 1$) ed é fatta !!!

Esercizio 3.[11 punti]

Sia L una lista ORDINATA di n numeri interi. Ogni elemento della lista contiene un campo chiave e un puntatore ($next$) all'elemento successivo. Quanto costa ricercare un elemento in L ? Fornire lo pseudo-codice relativo alla ricerca.
Aggiungere opportunamente un puntatore ad ogni elemento della lista in modo tale da ridurre il tempo di ricerca a $O(\sqrt{n})$. Spiegare a parole (o fornire il relativo pseudo-codice) il funzionamento del nuovo algoritmo di ricerca. Giustificare le risposte.

Traccia Soluzione Esercizio 3

All'elemento i-esimo della lista aggiungiamo un puntatore (che chiameremo $sqrt$) che punta all'elemento in posizione $i + \lfloor\sqrt{n}\rfloor$ della lista. Se l'elemento in posizione $i + \lfloor\sqrt{n}\rfloor$ non esiste il nuovo puntatore conterrà nil. La ricerca inizialmente userà il puntatore $sqrt$ per fare salti in avanti lunghi $\lfloor\sqrt{n}\rfloor$. Quando superiamo il valore ricercato torniamo indietro di un passo (dobbiamo quindi memorizzare due puntatori contemporaneamente) e procediamo con una ricerca (al massimo lunga $\lfloor\sqrt{n}\rfloor$ elementi) usando il puntatore $next$. Il tutto può costare al massimo $O(\sqrt{n})$ operazioni.