1. Esercizi di programmazione

Esercizio 1.1 – Cerca duplicati

Dato un vettore di n interi, scrivere un algoritmo che restituisca vero se un elemento compare due volte. Determinare una limitazione inferiore e una superiore alla complessità di questo problema.

Esercizio 1.2 – Cerca la coppia

Dato un array $A[1 \dots n]$ di interi e un intero v, scrivere un algoritmo che determini se esistono due elementi in A la cui somma sia esattamente v.

Esercizio 1.3 – Cerca la coppia – versione con 17

Dato un array $A[1 \dots n]$ di interi e un intero v, scrivere un algoritmo che determini se esistono due elementi in A la cui somma sia esattamente 17.

2. Ricorrenze

Esercizio 2.1 – Algoritmo di selezione deterministico

Si consideri la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 11/5n + T(\lfloor n/5 \rfloor) + T(\lfloor 7n/10 \rfloor) & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Individuare limiti inferiori e superiori tramite il metodo di sostituzione.

Esercizio 2.2 – Cicli FOR!

Si dimostri, per induzione, che $\sum_{i=1}^{n} i^h$ è $O(n^{h+1})$.

4. Alberi

Esercizio 3.1 - Grado di sbilanciamento

Dato un albero binario T, il *grado di sbilanciamento* di un nodo v è pari alla differenza, in valore assoluto, fra il numero di *foglie* presenti nel sottoalbero sinistro di v e il numero di *foglie* presenti nel sottoalbero destro di v. Il grado di sbilanciamento dell'albero T è pari al massimo grado di sbilanciamento dei nodi di T.
Scrivere un algoritmo che dato un albero T restituisce il grado di sbilanciamento dell'albero. Discuterne correttezza e complessità.

Esercizio 3.2 – Cammino radice–discendente

Dato un albero binario contenente interi, scrivere un algoritmo che restituisca la lunghezza del più lungo cammino monotono crescente radice-discendente, dove il discendente non è necessariamente foglia; con lunghezza si intende il numero totale di *archi* attraversati; e con monotona crescente si intende che i valori contenuti nei nodi della sequenza devono essere ordinati in senso crescente da radice a discendente. Discuterne correttezza e complessità.

4. Alberi + ricorrenze

Esercizio 4.1 – Alberi strutturalmente diversi

Due alberi binari si dicono "strutturalmente" diversi se disegnando correttamente i figli destri e sinistri, si ottengono figure diverse. Ad esempio, un nodo radice con un figlio destro è diverso da un nodo radice con un figlio sinistro.

1. Si dica quanti sono i possibili alberi binari strutturalmente diversi composti da 1, 2, 3 e 4 nodi.
2. Dare una formula di ricorrenza per il caso generale di n nodi.
3. Data la ricorrenza al punto 2) trovare il più stretto limite asintotico inferiore che riuscite a trovare.

Spoiler alert!

Ricorrenze

$$T(0) = 1 \leq c \cdot 0 \Leftrightarrow 1 \leq 0 \qquad \text{Falso!}$$
$$T(1) = 1 \leq c \cdot 1 \Leftrightarrow c \geq 1$$
$$T(2) = 22/5 + T(\lfloor 2/5 \rfloor) + T(\lfloor 14/10 \rfloor) = 22/5 + T(0) + T(1) = 32/5 \leq$$
$$T(3) = 33/5 + T(\lfloor 3/5 \rfloor) + T(\lfloor 21/10 \rfloor) = 33/5 + T(0) + T(2) = 70/5 \leq$$
$$T(4) = 44/5 + T(\lfloor 4/5 \rfloor) + T(\lfloor 28/10 \rfloor) = 44/5 + T(0) + T(2) = 81/5 \leq$$
$$T(5) = 55/5 + T(\lfloor 5/5 \rfloor) + T(\lfloor 35/10 \rfloor) = 55/5 + T(1) + T(3)$$

$$\begin{aligned} T(n) &\leq 11/5n + c\lfloor n/5 \rfloor + c\lfloor 7n/10 \rfloor \\ &\leq 11/5n + 1/5cn + 7/10cn \\ &= 9/10cn + 22/10n \leq cn \end{aligned}$$

Sommatorie

- Caso base ($h = 0$):

$$\sum_{i=1}^{n} i^0 = \sum_{i=1}^{n} 1 = n = n^{h+1}$$

- Passo induttivo. Supponiamo che la proprietà sia vera per ogni $k < h$; vogliamo dimostrare che la proprietà è vera per h:

$$\sum_{i=1}^{n} i^h = \sum_{i=1}^{n} i^{h-1} i \leq \sum_{i=1}^{n} i^{h-1} n = n \sum_{i=1}^{n} i^{h-1} = nO(n^h) = O(n^{h+1})$$

Alberi - Grado di sbilanciamento

integer, **integer** unbalance(TREE T)

if $T =$ **nil then**
  return $(0, 0)$
if $T.left =$ **nil and** $T.right =$ **nil then**
  return $(1, 0)$
$L_{leafs}, L_{max} \leftarrow$ unbalance($T.left$)
$R_{leafs}, R_{max} \leftarrow$ unbalance($T.right$)
return $(L_{leafs} + R_{leafs}, \mathsf{max}(L_{max}, R_{max}, |L_{leafs} - R_{leafs}|))$

Alberi – Percorso cammino-discendente

```
integer maxdepth(Tree T)

if T ≠ nil and (T.parent() = nil or T.parent().value < T.value) then
    return 1 + max(maxdepth(T.left()), maxdepth(T.right()))
else
    return −1
```

$$P(n) = \begin{cases} \sum_{k=0}^{n-1} P(k)P(n-1-k) & n > 1 \\ 1 & n \leq 1 \end{cases}$$