# Introduzione agli Algoritmi
# Esame Scritto a canali unificati

docenti: T. Calamoneri, A. Monti
Sapienza Università di Roma
13 Luglio 2021

**Esercizio 1 (10 punti):**
Si consideri la seguente funzione:

```
funzione Exam(n):
    if n <= 2:  return 2 * n;
    b ← n/2;
    tot ← n * n;
    for i = 1 to n:
         for j = 1 to i:
              tot ← tot + i − j;
    for i = 1 to 4:
         for j = 1 to 4:
             if i = j:   tot ← tot+ Exam(b)
             else:   tot ← tot + i − j;
    return tot.
```

a) Si imposti la relazione di ricorrenza che ne definisce il tempo di esecuzione giustificando l'equazione ottenuta.

b) Si risolva l'equazione usando il metodo iterativo, commentando opportunamente i passaggi del calcolo;

c) Si risolva l'equazione usando il metodo principale, specificando quale caso del teorema si applica e perché oppure per quale motivo non si può applicare il teorema.

**Esercizio 2 (10 punti):**

Progettare un algoritmo che, dato un array $A$ di $n$ interi distinti i cui elementi sono all'inizio in ordine crescente e da un certo punto in poi in ordine decrescente, restituisce in tempo $O(\log n)$ il massimo intero presente nell'array. Ad esempio: per $A = [8, 10, 20, 80, 100, 200, 400, 500, 3, 2, 1]$ l'algoritmo deve restituire il valore 500.

Dell'algoritmo proposto

a) si dia la descrizione a parole;

b) si scriva lo pseudocodice;

c) si calcoli il costo computazionale.

**Esercizio 3 (10 punti):**

Dato un albero binario $T$, radicato e con $n$ nodi, definiamo un nodo $u$ di $T$ *equilibrato* se il sottoalbero sinistro di $u$ e il sottoalbero destro di $u$ hanno entrambi lo stesso numero di nodi.

Progettare un algoritmo che, dato il puntatore $r$ alla radice di un albero binario memorizzato tramite record e puntatori, restituisca in tempo $O(n)$ il numero dei suoi nodi equilibrati.
Dell'algoritmo proposto

a) si dia la descrizione a parole;

b) si scriva lo pseudocodice;

c) si motivi il costo computazionale.

Qual è il numero minimo e qual è il numero massimo di nodi equilibrati che l'albero $T$ può avere? Motivare la risposta.

**Idee per la soluzione**

ESERCIZIO 1:

a) La relazione di ricorrenza è $T(n) = 4T\left(\frac{n}{2}\right) + \Theta\left(n^2\right)$ con $T(1) = \Theta(1)$. Infatti per ogni chiamata iterativa: il costo dei primi due for annidati risulta $\Theta\left(\sum_{i=1}^{n} i\right) = \Theta\left(n^2\right)$ mentre il costo degli ultimi due for annidati risulta $4 \cdot T(b) + \Theta(1) = 4 \cdot T\left(\frac{n}{2}\right) + \Theta(1)$

b) *il metodo iterativo.* Con un passo iterativo abbiamo

$$\begin{aligned} T(n) &= 4T\left(\frac{n}{2}\right) + \Theta\left(n^2\right) \\ &= 4\left(4T\left(\frac{n}{2^2}\right) + \Theta\left(\left(\frac{n}{2}\right)^2\right)\right) + \Theta\left(n^2\right) \\ &= 4^2 T\left(\frac{n}{2^2}\right) + \Theta(n^2) + \Theta(n^2) \\ &= 4^2 T\left(\frac{n}{2^2}\right) + 2 \cdot \Theta(n^2) \end{aligned}$$

Dopo $k$ iterazioni ottengo $4^k \cdot T\left(\frac{n}{2^k}\right) + k \cdot \Theta(n^2)$. Vado avanti finché $\frac{n}{2^k} = 1$ cioè $k = \log_2 n$. Ottengo quindi

$$4^{\log_2 n} \cdot T(1) + \log_2 n \cdot \Theta(n^2) = n^2 \cdot \Theta(1) + \Theta \cdot (n^2 \log n) = \Theta(n^2 \log n).$$

c) *il metodo principale:* abbiamo $a = 4$ e $b = 2$ e quindi $\Theta\left(n^{\log_b a}\right) = \Theta(n^2) = f(n)$. Siamo dunque nel secondo caso del teorema pertanto la soluzione dell'equazione è $\Theta(n^2 \log n)$.

ESERCIZIO 2:

a) Se il vettore contiene 1 o 2 elementi risolvo il problema direttamente. In caso contrario applico la ricerca binaria: sia $m$ la posizione di mezzo dell'array, se $A[m-1] < A[m]$ cerco nel sottoarray di destra che parte dalla posizione $m$ in caso contrario cerco nel sottoarray di sinistra che termina nella posizione $A[m-1]$.

b) Un possibile codice in python :

```
def esame2(A):
    i = 0
    j = len(A) - 1
```

```
while True:
    if j == i:  return A[i]
    if j == i + 1:  return max(A[i],  A[i + 1])
    m = (i + j)//2
    if  A[m − 1] < A[m] : i = m
    else:   j = m − 1
```

c) Ad ogni passo la dimensione dell'array in cui ricercare il valore massimo si dimezza di conseguenza il numero di iterazioni per un array di dimensione $n$ risulta $\log n$.

ESERCIZIO 3:

a) effettuo una visita postorder dell'albero. Ogni nodo restituisce al padre il numero di nodi presenti nel suo sottoalbero e il numero di nodi equilibrati presenti nel suo sottoalbero. Le foglie ovviamente restituiscono i valori 0 e 0.
Ciascun nodo interno ricevendo queste informazioni è in grado di trasmettere a sua volta le giuste informazioni al padre: aggiungendo 1 alla somma dei nodi del sottoalbero sinistro e del sottoalbero destro può calcolare il numero di nodi del suo sottoalbero inoltre il numero di nodi equilibrati nel suo sottoalbero è dato dalla somma dei nodi equilibrati del sottoalbero sinistro più i nodi equilibrati del sottoalbero destro più eventualmente 1 se il nodo stesso è equilibrato (vale a dire se figlio destro e figlio sinistro hanno restituito lo stesso numero di nodi). Il numero di nodi equilibrati restituito dalla radice è la soluzione al problema.

b) Un possibile codice in python :

```
def esame3(r):
  if r == None:
      return 0, 0
  equilibratiS, nodiS =esame3(r.left)
  equilibratiD, nodiD =esame3(r.right)
  equilibrati = equilibratiS + equilibratiD
  if nodiS == nodiD:   equilibrati+ = 1
  return equilibrati, nodiS + nodiD + 1
```

Si noti che, sebbene meno elegante, sarebbe stato corretto anche l'uso di una variabile globale per memorizzare il numero dei nodi equilibrati; in

questo caso, la funzione sarebbe stata identica a quella fatta a lezione per calcolare il numero dei nodi di un albero, con la sola aggiunta dell'aggiornamento della variabile globale in caso di ritorno di valori uguali dai figli di ciascun nodo. Si sottolinea che, invece, NON è una buona soluzione quella che utilizza due diverse funzioni ricorsive, una che calcola i nodi di un albero e una che ne calcola i nodi equilibrati, in cui la prima richiama la seconda poiché, evidentemente, questa soluzione non ha costo lineare bensí quadratico.

c) il costo è quello di una semplice visita dell'albero, quindi $O(n)$.

Il valore massimo è $n$ (se l'albero è binario completo tutti i suoi nodi sono equilibrati).
Il valore minimo è 1. Considera infatti una catena (un albero radicato in cui tutti i nodi interni hanno un solo figlio). Nella catena l'unico nodo equilibrato è la sua unica foglia. Nota che non può esistere un albero con 0 nodi equilibrati in quanto le foglie di un albero sono sempre bilanciate e ogni albero ha almeno una foglia (a meno che non si sia deciso esplicitamente di escludere le foglie dal conteggio).