

Es. 1. Si dimostri, utilizzando la definizione di Θ , che

$$f(n) = \lg^2(2n) + \lg 4n = \Theta(\lg^2 n)$$

mettendo in evidenza e commentando con chiarezza i passi seguiti.

Sol. Per dimostrare che $f(n)$ è in $\Theta(\lg^2 n)$ dobbiamo trovare tre costanti positive c, c' e n' tali che $c \lg^2 n \leq \lg^2(2n) + \lg 4n \leq c' \lg^2 n$ per ogni $n \geq n'$. Trattiamo le due disuguaglianze separatamente.

1. $c \lg^2 n \leq \lg^2(2n) + \lg 4n$ è banalmente vero per $c = 1$, perchè il logaritmo è crescente per ogni $n \geq 1$.

2. $\lg^2(2n) + \lg 4n \leq c' \lg^2 n$

Poichè $(\lg 2 + \lg n)^2 + \lg 4 + \lg n =$

$1 + \lg^2 n + 2 \lg n + 2 + \lg n =$

$\lg^2 n + 3 \lg n + 3$

la disuguaglianza diventa

$\lg^2 n + 3 \lg n + 3 \leq c' \lg^2 n$ che è vera per $c' = 7$ e $n \geq 2$.

Concludiamo che l'affermazione è vera per $c=1$, $c'=7$ e $n' = 2$.

Es.2. Si imposti la relazione di ricorrenza che definisce il tempo di esecuzione della seguente funzione e la si risolva usando il metodo della sostituzione. Si commentino opportunamente i passaggi del calcolo, si disegni l'albero della ricorsione e come si giunge alla previsione sull'andamento del tempo di calcolo, si imposti l'induzione con chiarezza, sia nello scrivere quanto si vuole dimostrare sia nel formulare l'ipotesi induttiva.

Strano(A,i,j)

{ $n = j-i+1$

if ($n \leq 1$) **return** 1;

$m = n/2$

while $n > 0$ **do** $n = n/2$

return Strano(A,i,i+m) + Strano(A,i+m,j)

}

La funzione è chiamata 2 volte su $n/2$ elementi e il ciclo while è eseguito in tempo logaritmico, quindi la relazione di ricorrenza è

$$T(n) = 2T(n/2) + \Theta(\lg n).$$

Si deve quindi risolvere la ricorrenza

$$T(n) = 2T(n/2) + c \lg n \text{ se } n \geq 1$$

$$T(n) = d \text{ altrimenti}$$

L'albero della ricorsione per fare una previsione sull'andamento è qui abbozzato:

$$\begin{array}{ccccccc}
 & & & & & & c \lg n \\
 & & & & & & / \quad \backslash \\
 & & & & & & c \lg n/2 \quad c \lg n/2 \\
 & & & & & & / \quad \backslash \quad / \quad \backslash \\
 & & & & & & c \lg n/2^2 \quad c \lg n/2^2 \quad c \lg n/2^2 \quad c \lg n/2^2
 \end{array}$$

Ponendo $n = 2^h$, l'albero ha altezza h , ogni nodo ha 2 figli e al livello i ogni nodo è etichettato con $c \lg n/2^i$. Quindi, sommando su ogni livello si ha $2^i c \lg n/2^i$ da cui:

$$\begin{aligned}
 T(n) &= 2^h d + \sum_{i=0 \dots h-1} 2^i c \lg n/2^i \\
 &= 2^h d + \sum_{i=0 \dots h-1} 2^i c \lg n/2^i = 2^h d + \sum_{i=0 \dots h-1} 2^i c (\lg n - i) \leq 2^h d + \sum_{i=0 \dots h-1} 2^i c \lg n \\
 &= 2^h d + c \lg n (2^h - 1) = O(n \lg n)
 \end{aligned}$$

Abbiamo maggiorato e il termine $c n \lg n$ è dominante rispetto a nd , quindi $T(n) = O(n \lg n)$.

Ora verifichiamo la previsione per induzione.

Faremo vedere che esistono due costanti positive k e n' tali che $T(n) \leq k n \lg n$, per ogni $n \geq n'$.

Sia vero per ipotesi induttiva per ogni $m < n$, consideriamo $T(n)$:

$$\begin{aligned}
 T(n) &= 2T(n/2) + c \lg n \leq \\
 &2k (n/2) \lg n/2 + c \lg n = \\
 &2k (n/2) (\lg n - 1) + c \lg n = \\
 &k \lg n - kn + c \lg n.
 \end{aligned}$$

Vediamo se troviamo due valori per k e n' tali che

$$k \lg n - kn + c \lg n \leq k n \lg n \Leftrightarrow -kn + c \lg n \leq 0 \Leftrightarrow c \lg n \leq kn \text{ quest'ultima disuguaglianza è vera per } k \geq c \text{ e per ogni } n \geq 1$$

Considerando il caso base abbiamo che $T(1) = d$ e $T(2) = 4d+c$ e $T(2) \leq 4k$ è vero prendendo $k = d+c$.

Es. 3. Si progetti un algoritmo ricorsivo che verifica se un albero binario T ha una foglia di profondità k . Si descriva a parole l'idea algoritmica, si analizzi il tempo di esecuzione asintotico, specificando caso peggiore e caso migliore, e si produca lo pseudocodice.

Esempio: se $k = 2$ la risposta è 1 (vero), ci sono 3 foglie di profondità 2. Se $k = 1$ allora la risposta è 0 (falso) perché non ci sono foglie di profondità 1.

```
      10
     50  5
    6   12 20
```

P.S. Soluzioni ridotte al solo pseudocodice non saranno corrette.

Sol. Modifichiamo la visita in preordine in modo da avere come parametro anche k oltre al nodo correntemente visitato e da controllare su ogni nodo se si tratta di una foglia della profondità voluta. Il parametro k deve essere decremento ad ogni discesa da padre a figlio, in modo che assumerà il valore 0 quando si trova la foglia di profondità k .

Si tratta di una variante della visita ma una volta trovata una foglia di profondità k la visita si ferma, quindi nel caso migliore si ha un tempo di esecuzione in $\Theta(h)$, quando la foglia è proprio sul cammino più a sinistra e nel caso peggiore è in $\Theta(n)$ quando la foglia cercata è invece sul cammino più a destra. In generale dunque l'algoritmo termina in $O(n)$.

Pseudocodice:

DepthLeaf(T, k)

Input: T è un puntatore alla radica di un albero binario

Output: vero se in T è presente una foglia di profondità k

if $T == \text{NULL}$ then return 0

if T è una foglia

if $k == 0$ return 1 else return 0

return DepthLeaf($T.\text{left}, k-1$) or LeftmostLeaf($T.\text{right}, k-1$)

Qui sfruttiamo l'implementazione dell'or logico, per cui se il primo dei due termini valuta a vero allora l'altro non è valutato.