DIE TI. UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria Informatica

CORSO DI
ALGORITMI E STRUTTURE DATI

VITO ROMANO
PAOLO RUSSO

Homeworks set #1

Esercizio 1.1. Notazione asintotica e crescita delle funzioni

Per ogni gruppo di funzioni, ordina le funzioni in ordine crescente di complessità asintotica (big-O):

1)
f1(n) = $n^{0.999999} \log n$
f2(n) = 10000000n
f3(n) = 1.000001^n
f4(n) = n^2

Notiamo che f1(n) cresce in maniera asintotica ed è più lenta di f2(n) , quindi f1(n)< f2(n) poiché per ogni c>0 risulta che $\log \log(n) = O(n^c)$. Quindi in base a questa considerazione, considerando c=0.000001, possiamo dedurre:

$$f1(n) = O(n^{0.999999} * n^{0.000001}) = O(n) = O(f2(n))$$

La funzione f2(n) è lineare mentre f4(n) risulta quadratica, pertanto, f2(n) = O(f4(n)).

Infine, notiamo che f3(n) risulta esponenziale (per n piccoli f3(n) ≈1 mentre per n->∞ f3(n) è un esponenziale), pertanto cresce molto più velocemente di f4(n) che è quadratica. Quindi, f4(n) =O(f3(n)).

L'ordine di queste funzioni è il seguente: **f1(n)<f2(n)<f4(n)<f3(n).**

2)
f1(n) = $2^{2^{1000000}}$
f2(n) = 2^{10000n}
f3(n) = $\binom{n}{2}$
f4(n) = $n\sqrt{n}$

Per quanto riguarda f1(n), nonostante gli esponenziali multipli, viene classificata come costante. f1(n) è asintoticamente più piccolo di f4(n) che cresce al variare di n. Per rendere ciò più visibile, possiamo scrivere f4(n) come $n^{\frac{3}{2}}$. Per quanto riguarda f3(n) deriva dalla formula di Gauss che è $O(n^2)$.

$(\frac{n!}{2*(n-2)!} = \frac{n*(n-1)*(n-2)!}{2*(n-2)!} = \frac{n*(n-1)}{2} = O(n^2)\ )$

Abbiamo quindi che f4(n)=O(f3(n)). Infine, poiché f3(n) è quadratica e f2(n) esponenziale risulta che f3(n)=O(f2(n)).

L'ordine di queste funzioni è il seguente: **f1(n)<f4(n)<f3(n)<f2(n).**

3)
f1(n) = $n^{\sqrt{n}}$
f2(n) = 2^n
f3(n) = $n^{10} \cdot 2^{n/2}$
f4(n) = $\sum_{i=1}^{n}(i+1)$

Andando a trasformare f4(n), f3(n) e f1(n).

Per quanto riguarda f4(n) lo posso scrivere come

$$\sum_{i=1}^{n}(i+1) = 1+2+3+\ \ldots\ldots\ = \frac{n*(n+1)}{2} - 1 = O(n^2)$$

f1(n)=$n^{\sqrt{n}} = 2^{\sqrt{n}*log\ (n)}$ mentre

f3(n)=$n^{10} * 2^{\left(\frac{n}{2}\right)} = 2^{loglog\ (n^{10})} * 2^{\left(\frac{n}{2}\right)} = 2^{\left(\frac{n}{2}\right)+10*log\ (n)}$

In base a queste considerazioni possiamo constatare che valutando f1(n) e f2(n) in termine di esponente risulta che in f3(n) presenta una funzione che cresce linearmente al crescere di n, a differenza dell'esponente di f1(n) che cresce più lentamente. Quindi f1(n)=O(f3(n)). E quindi vale anche f4(n)=O(f1(n)). Confrontato f3(n) con f2(n) risulta che f3(n)=O(f2(n)). Infatti, f3(n) e f2(n) nonostante hanno una funzione lineare nel loro esponente asintoticamente si comportano in maniera diversa. L'ordine di queste funzioni è il seguente: **f4(n)<f1(n)<f3(n)<f2(n).**

Esercizio 1.2. Notazione asintotica e crescita delle funzioni

Per ognuna delle seguenti funzioni si determini se f(n) = O(g(n)), g(n) = O(f(n)) o entrambe.

- $f(n) = \frac{(n^2-n)}{2}$ **ed** $g(n) = 6n$

  Vale che g(n) = O(f(n)) poiché $6n \leq c(n^2) \rightarrow 6 \leq cn$.

  Non vale il viceversa (f(n) = O(g(n))).

- $f(n) = n + 2\sqrt{n}$ **ed** $g(n) = n^2$

  Vale che f(n) = O(g(n)) poiché $n + 2\sqrt{n} \leq cn^2 \rightarrow \frac{1}{n} + \frac{2}{\sqrt{n}} \leq c$.

  Non vale il viceversa(g(n) = O(f(n))).

- $f(n) = nlog(n)$ **ed** $g(n) = n\frac{\sqrt{n}}{2}$

  Vale che f(n) = O(g(n)) poiché $nlog(n) \leq cn^{\frac{3}{2}} \rightarrow log(n) \leq c\sqrt{n}$

  Non vale il viceversa(g(n) = O(f(n))).

- $f(n) = n + log(n)$ **ed** $g(n) = \sqrt{n}$

  Vale che g(n) = O(f(n)) poiché $\sqrt{n} \leq cn \rightarrow \frac{1}{\sqrt{n}} \leq c$.

  Non vale il viceversa (f(n) = O(g(n))).

- $f(n) = 2(log(n))^2$ **ed** $g(n) = log(n) + 1$

  Vale che g(n) = O(f(n)) poiché $log(n) \leq clog^2(n) \rightarrow \frac{1}{log(n)} \leq c$.

  Non vale il viceversa (f(n) = O(g(n))).

- $f(n) = 4nlog(n) + n$ **ed** $g(n) = \frac{(n^2-n)}{2}$

  Vale che f(n) = O(g(n)) poiché $nlog(n) \leq cn^2 \rightarrow \frac{log(n)}{n} \leq c.$
  Non vale il viceversa (g(n) = O(f(n))).

Esercizio 1.3. Notazione asintotica e crescita delle funzioni

Si indichi se le seguenti affermazioni sono vere o false

- 2^{n+1}=O(2^n) $\rightarrow 2^n * 2 = O(2^n) \rightarrow 2^n * 2 \leq c * 2^n \rightarrow$ c ≥ 2 , la condizione è sempre verificata, quindi VERA.

- 2^{2n}=O(2^n) $\rightarrow 2^{2n} \leq c * 2^n \rightarrow$ c≥ $2^{2n-n} \rightarrow$ c ≥ 2^n, la condizione non è mai verificata, quindi FALSA.

Esercizio 1.4. Ricorrenze
Fornire il limite inferiore e superiore per *T(n)* nella seguente ricorrenza, usando il metodo dell'albero delle ricorrenze ed il teorema dell'esperto se applicabile. Si fornisca il limite più stretto possibile giustificando la risposta.

- $T(n) = 2T\left(\frac{n}{2}\right) + O(\sqrt{n}\,)$

  Usando il metodo dell'esperto risulta che a=2, b=2 $\rightarrow n^{log_b a} = n$,

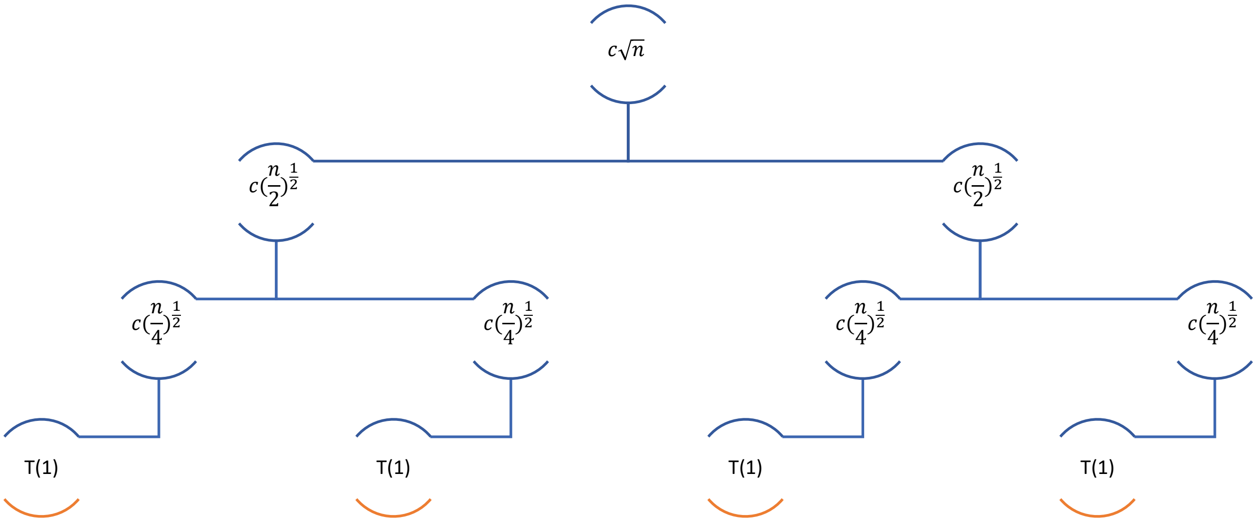
  con $f(n) = O(\sqrt{n})$.

  Dal primo caso del teorema dell'esperto risulta che

  $O(\sqrt{n}) = O(n^{1-\epsilon})\ con \quad \epsilon = \frac{1}{2}$

  Quindi $T(n) = \theta(n)$.

  Usando il metodo dell'albero di ricorrenza:



$$T(n) = 2T\left(\frac{n}{2}\right) + c\sqrt{n}$$

l'ultimo livello si avrà per:

$$\frac{n}{2^i} = 1 \Rightarrow i = \log_2 n$$

Quindi:

$$T(n) = \sum_{i=0}^{\log_2 n - 1} 2^i c\left(\frac{n}{2^i}\right)^{\frac{1}{2}} + 2^{\log_2 n} T(1) =$$

$$= \sum_{i=0}^{\log_2 n - 1} \sqrt{2^i}\, c\sqrt{n} + \Theta\left(n^{\log_2 2}\right) < \sum_{i=0}^{\infty} \sqrt{2^i}\, c\sqrt{n} + \Theta(n)$$

$$\Rightarrow \frac{c\sqrt{n}}{\sqrt{2} - 1} + \Theta(n) \Rightarrow T(n) = \Theta(n)$$

- $T(n) = T(\sqrt{n}) + \Theta(log(log(n)))$

Applicando il principio di sostituzione risulta che $log\,(n) = m$ **da cui risulta che** $n = 2^m$.

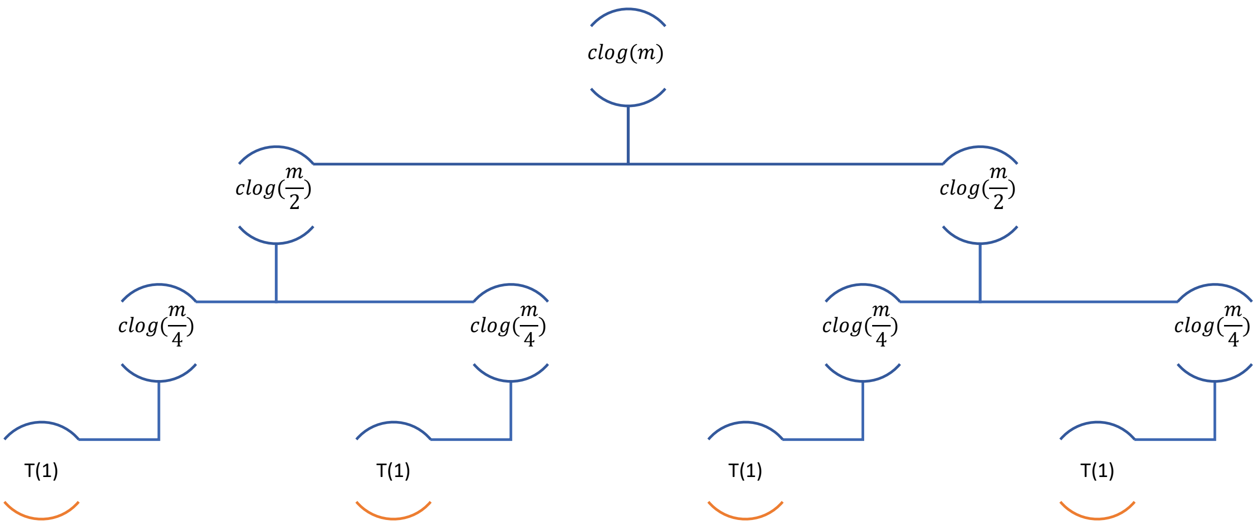
Dalla sostituzione otteniamo che: $T(2^m) = T\left(2^{\frac{m}{2}}\right) + \Theta(log\,(m))$

Impossibile applicare il metodo dell'esperto perché:

$a = 1,\ b = 2 \Rightarrow n^{\log_b a} = 1$;

$$\log_n \neq \begin{cases} O(n^{1-\varepsilon}) \\ \Theta(1) \\ \Omega(n^{1+\varepsilon}) \end{cases}$$

Usando il metodo dell'albero di ricorrenze:



$$S(m) = S\left(\frac{m}{2}\right) + c \log m$$

l'ultimo livello si avrà per:

$$\frac{n}{i} = 1 \Rightarrow i = \log n$$

Quindi:

$$T(m) = \sum_{i=0}^{\log m - 1} c \log\left(\frac{m}{2^i}\right) + 1^{\log m}\, T(1) =$$

$$= \sum_{i=0}^{\log m - 1} c\left[\log m - \log 2^i\right] + \Theta(1) =$$

$$= c \log m \sum_{i=0}^{\log m - 1} 1 - c \sum_{i=0}^{\log m - 1} i < c \log m \sum_{i=0}^{\infty} 1 - c \sum_{i=0}^{\infty} i \Rightarrow$$

$$\Rightarrow c \log m (\log m) - c\left(\frac{1}{2}\log^2 m - \frac{1}{2}\log m\right) \Rightarrow$$

$$\Rightarrow c \log^2 m - \frac{c \log^2 m}{2} - \frac{c}{2}\log m \Rightarrow \frac{c}{2}\log^2 m - \frac{c}{2}\log m \Rightarrow$$

$$\Rightarrow S(m) = \Theta(\log^2(m)) \Rightarrow T(n) = \Theta(\log^2(\log n))$$

- $T(n) = 10T\left(\frac{n}{3}\right) + 17n^{1.2}$

Usando il metodo dell'esperto risulta che a=3, b=10 $\rightarrow n^{log_b a} = n^{2.09}$,

con $f(n) = O(17n^{1.2})$.

Dal primo caso del teorema dell'esperto risulta che

$O(n^{1.2}) = O(n^{2.09-\epsilon})\ con \quad \epsilon = 0.89$

Quindi $T(n) = \theta(n^{log_3 10})$.

Usando il metodo dell'albero di ricorrenza:

Calcolo l'albero di ricorrenza per:

$$T(n) = 10T\left(\frac{n}{3}\right) + cn^{1,2}$$

l'ultimo livello si avrà per:

$$\frac{n}{3^i} = 1 \Rightarrow i = \log_3 n$$

Quindi:

$$T(n) = \sum_{i=0}^{\log_3 n - 1} 10^i\, c\left(\frac{n}{3^i}\right)^{1,2} + 10^{\log_3 n}\, T(1) =$$

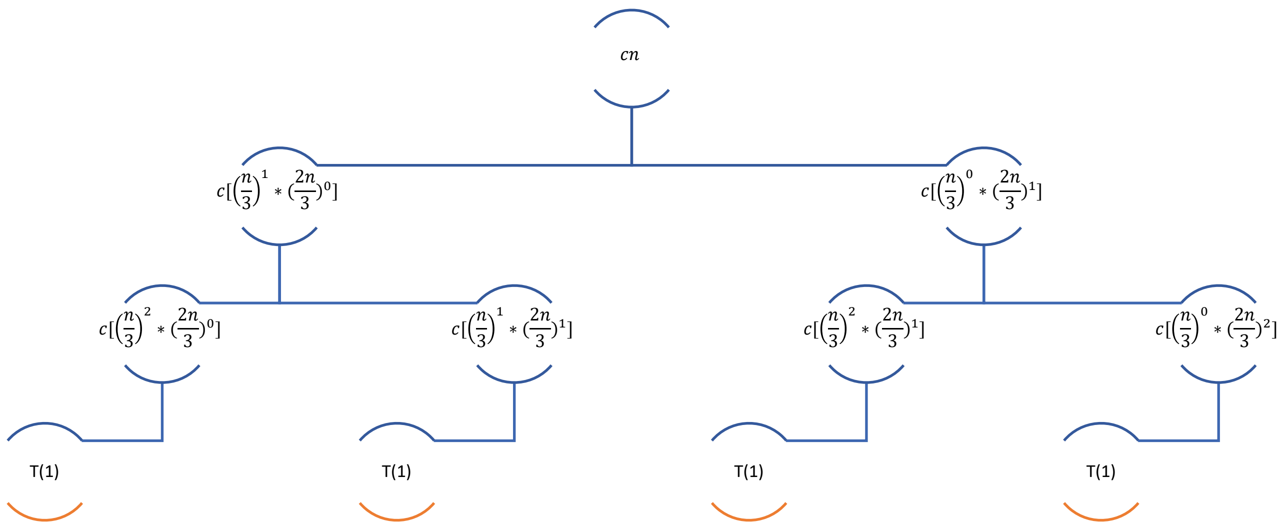
$$= \sum_{i=0}^{\log_3 n - 1} (2,676)^i\, cn^{1,2} + \theta\left(n^{\log_3 10}\right) = \left[\frac{(2,676)^{\log_3 n} - 1}{(2,676) - 1}\right] cn^{1,2} + \theta\left(n^{\log_3 10}\right)$$

$$\Rightarrow T(n) = \theta\left(n^{\log_3 10}\right)$$

L'albero di ricorrenza presenta dieci figli per ogni padre.

- Utilizzando l'albero di ricorsione dimostrate che la soluzione della ricorrenza $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$, dove c'è una costante è Ω(nlogn).

  Andando a descrivere il nostro albero di ricorsione, risulta:



Come possiamo constatare la somma dei valori presenti in ogni livello k associato all'albero di ricorsione è pari a cn.

Pertanto: $\sum_{i=0}^{k} cn * (\frac{1}{3} + \frac{2}{3})^k$ = cn.

Possiamo, constatare che il sottoalbero di destra sta facendo più lavoro del sottoalbero di sinistra, pertanto vale che $\frac{2n}{3} > \frac{n}{3}$, il caso peggiore riguarda il percorso che va dal nodo radice alla foglia

è il seguente $\frac{2}{3}n \rightarrow \left(\frac{2}{3}n\right)^2 \rightarrow ----1$.

Poiché $\left(\frac{2}{3}n\right)^k = 1$, quando k=$log_{\frac{3}{2}}(n)$, l'altezza dell'albero risulta $log_{\frac{3}{2}}(n)$. Non possiamo calcolare il costo di tutte le foglie in modo immediato perché non si tratta di un albero binario. Volendo dimostrare $\boldsymbol{\Omega}$(nlogn) ci concentriamo sul ramo più breve di sinistra, verificando il limite inferiore. L'altezza del ramo di sinistra è $log_3 n$.

quindi il costo dell'algoritmo per questo ramo sarà: T(n) = $cnlog_3 n$ = $\boldsymbol{\Omega}(n\ logn)$.