Corso di laurea in Informatica
Introduzione agli Algoritmi
Didattica blended

Esercizi per casa

SAPIENZA
UNIVERSITÀ DI ROMA

ESERCIZIO 1:

- Classifica le seguenti funzioni per ordine di crescita, vale a dire trova un ordinamento $g_1, g_2, \ldots$ delle funzioni che soddisfi $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$ ... .

- Partiziona la lista in classi di equivalenza di modo che le funzioni $f(n)$ e $g(n)$ sono nella stessa classe se e solo se $f(n) = \Theta(g(n))$.

$\left(\sqrt{2}\right)^{\lg n}$ n^2 $n!$ $\lg n^n$ n^3 $\lg^2 n$

$\lg(n!)$ 2^{2^n} $n^{\frac{1}{\lg n}}$ $\lg \lg n$ $n \cdot 2^n$ $n^{\lg \lg n}$ 5

$\lg n$ $2^{\lg n}$ $4^{\lg n}$ $(\lg n)^{\lg n}$ $2^{2^{n+1}}$ $(n+1)!$ e^n

$n \cdot lg\, n$ 2^n n $2^{\sqrt{2 \lg n}}$ $\left(\frac{3}{2}\right)^n$

le funzioni sono ordinate per ordine di crescita decrescente. Le funzioni nella stessa riga sono Θ l'una dell'altra

$$g_1(n) = 2^{2^{n+1}}$$
$$g_2(n) = 2^{2^n}$$
$$g_3(n) = (n+1)!$$
$$g_4(n) = n!$$
$$g_5(n) = e^n$$
$$g_6(n) = n2^n$$
$$g_7(n) = 2^n$$
$$g_8(n) = \left(\frac{3}{2}\right)^n$$

$$g_9(n) = n^{\log\log n}$$
$$g_{10} = (\lg n)^{\lg n}$$
$$g_{11}(n) = n^3$$
$$g_{12}(n) = n^2$$
$$g_{13}(n) = 4^{\lg n}$$
$$g_{14}(n) = n \lg n$$
$$g_{15}(n) = \lg n^n$$
$$g_{16}(n) = \lg(n!)$$
$$g_{17}(n) = 2^{\lg n}$$
$$g_{18}(n) = n$$
$$g_{19}(n) = \left(\sqrt{2}\right)^{\log n}$$
$$g_{20}(n) = 2^{\sqrt{2 \lg n}}$$
$$g_{21}(n) = \lg^2 n$$
$$g_{22}(n) = \lg n$$
$$g_{23}(n) = \lg\lg n$$
$$g_{24}(n) = 5$$
$$g_{25}(n) = n^{\frac{1}{\lg n}}$$

Per dimostrare che l'ordinamento proposto è quello giusto basta provare che per le due funzioni $g_i(n)$ e $g_{i+1}(n)$ presenti nella stessa riga vale $\lim_{n\to+\infty} \frac{g_i(n)}{g_{i+1}(n)} = c$

mentre se le due funzioni sono su righe successive allora $\lim_{n\to+\infty} \frac{g_i(n)}{g_{i+1}(n)} = +\infty$

$$g_1(n) = 2^{2^{n+1}}$$
$$g_2(n) = 2^{2^n}$$
$$g_3(n) = (n+1)!$$
$$g_4(n) = n!$$
$$g_5(n) = e^n$$
$$g_6(n) = n2^n$$
$$g_7(n) = 2^n$$
$$g_8(n) = \left(\frac{3}{2}\right)^n$$

$$\lim_{n\to+\infty} \frac{g_3(n)}{g_4(n)} = \lim_{n\to+\infty} \frac{(n+1)!}{n!} = \lim_{n\to+\infty} (n+1) = +\infty$$

$$\lim_{n\to+\infty} \frac{g_6(n)}{g_7(n)} = \lim_{n\to+\infty} \frac{n2^n}{2^n} = \lim_{n\to+\infty} n = = +\infty$$

$$\lim_{n\to+\infty} \frac{g_7(n)}{g_8(n)} = \lim_{n\to+\infty} \frac{2^n}{\left(\frac{3}{2}\right)^n} = \lim_{n\to+\infty} \left(\frac{4}{3}\right)^n = +\infty$$

Vale $\left(\frac{n}{2}\right)^{\frac{n}{2}} \leq n! \leq n^n$

infatti: $\left(\frac{n}{2}\right)^{\frac{n}{2}} < 1 \cdot 2 \ldots \cdot \left(\frac{n}{2}\right) \ldots \cdot n = n! = 1 \cdot 2 \ldots \cdot n < n^n$

$$\lim_{n\to+\infty} \frac{g_4(n)}{g_5(n)} = \lim_{n\to+\infty} \frac{n!}{e^n} \geq \lim_{n\to+\infty} \frac{\left(\frac{n}{2}\right)^{\frac{n}{2}}}{e^n} = \lim_{n\to+\infty} \left(\frac{n}{2\sqrt{e}}\right)^{\frac{n}{2}} = +\infty$$

$$\lim_{n\to+\infty} \frac{g_8(n)}{g_9(n)} = \lim_{n\to+\infty} \frac{\left(\frac{3}{2}\right)^n}{\lg(n!)} \geq \lim_{n\to+\infty} \frac{\left(\frac{3}{2}\right)^n}{\lg(n^n)} = \lim_{n\to+\infty} \frac{\left(\frac{3}{2}\right)^n}{n \lg n} = +\infty$$

$$\lim_{n\to+\infty} \frac{g_{16}(n)}{g_{17}(n)} = \lim_{n\to+\infty} \frac{\log(n!)}{2^{\log n}} \geq \lim_{n\to+\infty} \frac{\log\left(\frac{n}{2}\right)^{\frac{n}{2}}}{n} = \lim_{n\to+\infty} \frac{\frac{n}{2}\log\left(\frac{n}{2}\right)}{n} = \lim_{n\to+\infty} \frac{\log\left(\frac{n}{2}\right)}{2} = +\infty$$

ESERCIZIO 2:

- Calcolare l'andamento asintotico delle seguenti funzioni:

- $f_1(n) = 4^{\log n}$
- $f_2(n) = \left(\sqrt{2}\right)^{\log n}$
- $f_3(n) = 2^{\frac{\log n}{2}} + 5n$
- $f_4(n) = 3n \log n + 2n^2$

- $f_5(n) = 2^{\log \frac{n}{2}} + 5n$

segue ESERCIZIO 2:

- Calcolare l'andamento asintotico delle seguenti funzioni:

$$
\begin{aligned}
f_1(n) &= 4^{\log n} \\
&= 2^{2 \log n} \\
&= \left(2^{\log n}\right)^2 \\
&= n^2 \\
&= \Theta(n^2)
\end{aligned}
$$

$$
\begin{aligned}
f_2(n) &= \left(\sqrt{2}\right)^{\log n} \\
&= 2^{\frac{\log n}{2}} \\
&= n^{\frac{1}{2}} \\
&= \Theta\left(\sqrt{n}\right)
\end{aligned}
$$

segue ESERCIZIO 2:

- Calcolare l'andamento asintotico della seguente funzioni:

$$
\begin{aligned}
f_3(n) &= 2^{\log \frac{n}{2}} + 5n \\
&= 2^{\log n - 1} + 5n \\
&= \frac{2^{\log n}}{2} + 5n \\
&= \frac{n^2}{2} + 5n \\
&= \frac{11}{2} n \\
&= \Theta(n)
\end{aligned}
$$

segue ESERCIZIO 2:

- Calcolare l'andamento asintotico della funzione:

$$
\begin{aligned}
f_4(n) &= 3n \log\, n + 2n^2 \\
&\leq 3 \cdot n \cdot n + 2n^2 \\
&= 3n^2 + 2n^2 \\
&= 5n^2 \\
&= O(n^2)
\end{aligned}
$$

$$
\begin{aligned}
f_4(n) &= 3n \log\, n + 2n^2 \\
&\geq 2n^2 \\
&= \Omega(n^2)
\end{aligned}
$$

$$\Omega(n) = f_4(n) = O(n) \Longrightarrow f_4(n) = \Theta(n)$$

segue ESERCIZIO 2:

- Calcolare l'andamento asintotico delle seguenti funzioni:

$$
\begin{aligned}
f_5(n) &= 2^{\frac{\log n}{2}} + 5n \\
&= \sqrt{n} + 5n \\
&\leq 6n \\
&= O(n)
\end{aligned}
$$

$$
\begin{aligned}
f_5(n) &= 2^{\frac{\log n}{2}} + 5n \\
&= \sqrt{n} + 5n \\
&\geq 5n \\
&= \Omega(n)
\end{aligned}
$$

$$\Omega(n) = f_5(n) = O(n) \Longrightarrow f_5(n) = \Theta(n)$$

ESERCIZIO 3

Sia dato un vettore A di interi ordinato ed un valore a.
Il problema è quello di sapere quante occorrenze di a sono presenti in A (ovviamente la risposta sarà un intero tra 0 e la cardinalità del vettore).

- Si progetti un algoritmo per risolvere tale problema in tempo $\Theta(\log n)$.

Per l' algoritmo si descriva a parole l'idea, e poi si scriva lo pseudocodice.

segue ESERCIZIO 3:

cerco la posizione della prima occorrenza di a con una ricerca binaria modificata (`ricerca_binaria_primo(A,a)`) che restituisce $None$ se ha non c'è.

Se viene restituito $None$ (e quindi a non è presente nel vettore) l'algoritmo termina restituendo 0.

In caso contrario cerco la posizione dell'ultima occorrenza di a con la ricerca binaria modificata (`ricerca_binaria_ultimo(A,a)`) e restituisco alla fine il numero di elementi nell'intervallo individuato.

```
def Conta_occorrenze_in_Intervallo_ordinato(A,a):
    ind_a=ricerca_binaria_primo(A,a)
    if ind_a==None: return 0
    ind_b=ricerca_binaria_ultimo(A,b)
    return ind_b-ind_a+1
```

l'algoritmo richiede tempo ricerca_binaria(A,a)

segue ESERCIZIO 3:

`ricerca binaria_primo(A,v)`

procedura che in $O(\log n)$ restituisce la posizione in cui si trova la prima occorrenza di v nel vettore A . Restituisce $None$ se l'elemento non è presente.

```
def ricerca binaria_primo(A,v):
    a = 0
    b = len(A)-1
        while a <= b:
     m = (a+b)//2
    if A[m] == v:
         if m==0 or A[m-1]!=v:
             return m
         else:
             b=m-1
     elif A[m] > v:
          b = m - 1
    else:
        a = m + 1
return None
```

segue ESERCIZIO 3:

`ricerca binaria_ultimo(A,v)`

procedura che in $O(\log n)$ restituisce la posizione in cui si trova l'ultima occorrenza di v nel vettore A . Restituisce $None$ se l'elemento non è presente.

```
def ricerca binaria_ultimo(A,v):
    a = 0
    b = len(A)-1
        while a <= b:
     m = (a+b)//2
    if A[m] == v:
         if m==len(A)-1 or A[m+1]!=v:
             return m
         else:
             a=m+1
     elif A[m] > v:
          b = m - 1
    else:
        a = m + 1
return None
```