Sol Esercizio 1

Es. Notazione asintotica:

1. **Si dimostri che $f(n)+g(n) = \Theta(\max\{f(n),g(n)\})$ sotto l'ip. $f(n),g(n) >0$, a partire da un certo n_0.**

poiché $f(n) \leq \max\{f(n),g(n)\}$, e
$g(n) \leq \max\{f(n),g(n)\}$, sommando termine a termine:

$f(n)+g(n) \leq 2 * \max\{f(n),g(n)\})$, per ogni $n \geq n_0$ e questo vuole dire che $f(n)+g(n) = O(\max\{f(n),g(n)\})$
(qui la costante c è 2 e n_0 è quello della positività delle funzioni.)

poiché $f(n)+g(n) \geq \max\{f(n),g(n)\}$, per ogni $n \geq n_0$ e possiamo concludere che $f(n)+g(n) = \Omega(\max\{f(n),g(n)\})$.
(qui la costante c è 1 e n_0 è sempre quello della positività delle funzioni.)

Quindi possiamo concludere che $f(n)+g(n) = \Theta(\max\{f(n),g(n)\})$

Sol esercizio 2

Si dimostri che se $f(n) = \Theta(n^k)$, per una costante k, cioè f(n) è polinomiale di grado k, allora $lg(f(n)) = \Theta(lg\ n)$, dove $lg(x) = \log_2(x)$.

Se $f(n) = \Theta(n^k)$, questo vuol dire che esistono c',c" e n_0, tali che $c'n^k \leq f(n) \leq c"n^k$, per ogni $n \geq n_0$.

Prendendo la prima disuguaglianza $c'n^k \leq f(n)$ e applicando il logaritmo a entrambi i membri si ottiene $lg(c'n^k) \leq lg(f(n))$, visto che il logaritmo è una funzione crescente.

Si ottiene quindi $lg(c') + klg\ n \leq lg(f(n))$, e a maggior ragione $klg(n) \leq lg(f(n))$, per ogni $n \geq n_0$ e cioè $lg(f(n)) = \Omega(lg\ n)$.

Prendendo la seconda disuguaglianza, $f(n) \leq c"n^k$, applicando il logaritmo a entrambi i membri si ottiene $lg(f(n)) \leq lg\ (c"n^k)$, come prima applicando le regole del logaritmo si ottiene

$lg(f(n)) \leq lg\ (c") + k\ lg(n) \leq (lg\ (c") + k)lg(n)$, per ogni $n \geq n_0$ e cioè $lg(f(n)) = O(lg\ n)$.

Questo dimostra che $lg(f(n)) = \Theta(lg\ n)$ se $f(n) = \Theta(n^k)$.

Esercizi notazione asintotica 1

Si confronti nlg n con n^2 : nlg n = $O(n^2)$ o n^2 = O(nlgn)?

Sol.: nlg n = $O(n^2)$ e $n^2 \neq$ O(nlgn)

nlg n = $O(n^2)$, infatti esistono c ed n_0, tali che n lg n $\leq cn^2 \Leftrightarrow$ $cn^2 - n \lg n \geq 0 \Leftrightarrow$

$n(cn - \lg n) \geq 0 \Leftrightarrow$ $n \geq 0$ e $(cn - \lg n) \geq 0$, perchè sia vero che $cn \geq \lg n$ basta

prendere $n_0 = 2$ e $c = 1$.

$n^2 \neq$ O(nlgn), infatti se $n^2 = O(n \lg n)$ allora dovrebbero esistere d ed n'_0, tali che $n^2 \leq dn \lg n$, per ogni $n \geq n'_0$, ma per $n \geq 2$, $n^2 \leq dn \lg n$ equivale a dire che $n \leq d \lg n$, e per un certo d. Ma per ogni scelta di d troviamo un valore di n per cui è falso.

Se per ogni scelta di d si prende $n = 2^d$, allora dovrebbe essere $2^d \leq d^2$, cosa falsa per ogni d≥5.

Si confronti nlg n con n: nlgn = O(n) o n = O(nlgn) ?

Sol.: nlg n ≠ O(n) mentre n = O(nlgn)

Esercizi notazione asintotica 1

É vero che $n \lg n = O(n \lg^2 n)$?
Sol.: Si deve verificare se esistono $c \geq 0$ ed n_0 tali che $n \lg n \leq c n \lg^2 n$, per ogni $n \geq n_0$.

Se $n \geq 2$, possiamo dividere l'espressione per $n \lg n$ e ottenere $1 \leq c \lg n$, che è vero per esempio con $c = 1$ e ogni $n \geq 2$, quindi $n_0 = 2$.

É vero che $n \lg n^5 = O(n \lg n)$?
Sol.: Sì perché $n \lg n^5 = 5 n \lg n$

Esercizi notazione asintotica 2

$n^2 + n\lg n = O(\ldots..)$

Sol.: $n^2 + n\lg n = O(n^2)$

$n + n\lg n = O(\ldots..)$

Sol.: $n + n\lg n = O(n \lg n)$

$\lg n + \lg\lg n = O(\ldots.)$

$\lg n + \lg\lg n = O(\lg n)$

Es. Not. Asint. e Analisi algoritmi 1

Se un algoritmo ha tempo di esecuzione $\Theta(n^2)$ nel caso peggiore, posso dedurne che nel caso migliore terminerà in $\Theta(n^2)$ passi?

Es. Not. Asint. e Analisi algoritmi 1
Sol.

La risposta è no, perché non è detto che i due casi abbiano la stessa complessità, come nel caso dell'insertionSort in cui il caso peggiore è in $\Theta(n^2)$, ma il caso migliore è in $\Theta(n)$.

Es. Not. Asint. e Analisi algoritmi 2

Se si dimostra che un algoritmo ha tempo di esecuzione $\Omega(n^2)$ nel caso migliore, è possibile che in qualche caso l'algoritmo termini in O(n) passi?

Es. Not. Asint. e Analisi algoritmi 2 Sol.

Se si dimostra che un algoritmo ha tempo di esecuzione $\Omega(n^2)$ nel caso migliore, è possibile che in qualche caso l'algoritmo termini in O(n) passi?

La risposta è no. Se nel caso migliore si è dimostrato che il limite inferiore alla complessità è $\Omega(n^2)$, ogni altro caso ha questo stesso limite inferiore e quindi non può avere una complessità in O(n).

Es. Not. Asint. e Analisi algoritmi 3

Se si dimostra che un algoritmo ha tempo di esecuzione $\Omega(n^2)$ nel caso peggiore, è possibile che in qualche caso l'algoritmo termini in O(n) passi?

Es. Not. Asint. e Analisi algoritmi 3

Se si dimostra che un algoritmo ha tempo di esecuzione $\Omega(n^2)$ nel caso peggiore, è possibile che in qualche caso l'algoritmo termini in O(n) passi?

Qui la risposta è sì, perché il limite inferiore per il caso peggiore può non valere per il caso migliore, si può sempre prendere l'insertionSort come esempio.

Es. Not. Asint. e Analisi algoritmi 3

a.Si consideri un algoritmo in cui un array ordinato di n elementi viene modificato in modo tale che gli elementi che sono potenze di 2 vengono divisi per 2.

E' vero che nel caso peggiore l'algoritmo esegue un numero di divisioni pari a Θ(lg n)? Se sì, perché … Se no, perché …

Sol. No, nel caso peggiore tutti i numeri sono potenze di 2 e quindi si farebbero Θ(n) divisioni.

E' vero che nel caso migliore l'algoritmo esegue un numero di divisioni pari a Θ(1)? Se sì, perché … Se no, perché …

Sol. Sì, perchè il caso migliore si verifica se non ci sono potenze di 2 o ce n'è una sola.

Es. Not. Asint. e Analisi algoritmi 3

a.Si consideri un algoritmo in cui un array ordinato di n elementi viene modificato in modo tale che gli elementi che sono potenze di 2 vengono divisi per 2.

E' giusto dire che l'algoritmo esegue un numero di divisioni pari a O(n)? Se sì, perché … Se no, perché …

Sol. Sì perchè al più si fanno tante divisioni quanti sono gli elementi, come abbiamo visto nel caso peggiore

E' vero che nel caso peggiore l'algoritmo ha un tempo di esecuzione asintotico Θ(n)? Se sì, perché … Se no, perché …

Sol. Sì perchè si deve scorrere tutto l'array ed eseguire tante divisioni quanti sono gli elementi.

Es. Not. Asint. e Analisi algoritmi 3

Supponiamo di avere un algoritmo che è diviso in più passi:

il primo passo è eseguito in O(n lg n)

il secondo è eseguito in Θ(n) nel caso peggiore

il terzo consiste in un ciclo che viene eseguito n volte, e ogni esecuzione è in O(lg n).

Qual'è il migliore limite superiore che si può proporre per tutto l'algoritmo?

Sol. : O(n lg n)

Pseudocodice Bubblesort modificato

```
BubbleSort(A)
n = len(A)
flag= true
while flag
     flag = false
     for (j = 0; j ≤ n-2; j++)
          for k = n-1 downto j+1 do
          if A[k] < A[k-1] then
               flag = true
               scambia A[k] e A[k-1]
```

La modifica fa sì che in assenza di scambi si termini l'esecuzione.

Ulteriore modifica del bubbleSort

Potremmo modificare il bubbleSort facendo in modo da tenere traccia dell'ultimo scambio avvenuto in modo che al successivo scorrimento non si vada a esaminare la porzione di array oltre quel punto. In più dettaglio se l'ultimo scambio è avvenuto tra gli elementi di indice i e i+1 allora non si andrà a esaminare tutti gli elementi tra 0 e i.

Domanda 1: è corretto?