DIETI 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

Prof. ROBERTO PIETRANTUONO

Homeworks set #1

Istruzioni

Si prepari un file PDF riportante il vostro nome e cognome (massimo 2 studenti). Quando è richiesto di fornire un algoritmo, si intende scritto in pseudo-codice e riportato nel PDF. Laddove opportuno, si fornisca una breve descrizione della soluzione: l'obiettivo non è solo eseguire l'esercizio e riportare il risultato, ma far comprendere lo svolgimento.

Homeworks esercitativi

Esercizio 1.1. Notazione asintotica

Per ognuna delle seguenti affermazioni, si dica se essa è **sempre vera**, **mai vera**, o **a volte vera,** per funzioni asintoticamente non-negative. Se la si considera sempre vera o mai vera, si spieghi il perché. Se è a volte vera, si dia un esempio per cui è vera e uno per cui è falsa.

- $f(n) = O(f(n)^2)$
- $f(n) + O(f(n)) = \Theta(f(n))$
- $f(n) = \Omega(g(n))$ e $f(n) = o(g(n))$ - Nota la notazione *little-o*

Esercizio 1.2. Complessità.

Dimostrare che per qualsiasi costante reale a e b ,con $b > 0$, $(n+a)^b = \Theta(n^b)$

Esercizio 1.3. Complessità.

E' vero che $2^{n+1} = O(2^n)$? E' vero che $2^{2n} = O(2^n)$?

Esercizio 1.4. Ricorrenze

Fornire il limite inferiore e superiore per $T(n)$ nelle seguenti ricorrenze. Si assume che T(n) è costante per $n \leq 10$. Si fornisca il limite più stretto possibile giustificando la risposta (es: se T(n) è O(log$n$) essa è chiaramente anche O(n): la risposta dovrebbe essere O(log$n$)).

- $T(n)=2T(n/3)+n\lg n$
- $T(n) = 3T(n/5) + \lg^2 n$

Esercizio 1.5 Ricorrenze

Utilizzando l'albero di ricorsione, dimostrate che la soluzione della ricorrenza
$T(n) = T(n/3) + T(2n/3) + cn$, dove c è una costante, è $\Omega(n\log n)$

Homeworks di verifica

Homework 1.1. Inversioni

Sia A[1…n] un array di n numeri naturali distinti. Se i<j e A[i] > A[j], allora la coppia (i, j) è detta ***inversione*** di A.

- Elencate le cinque inversioni dell'array (2, 3, 8, 6 ,1)

- Quale array con elementi estratti dall'insieme {1, 2, … n} ha più inversioni? Quante inversioni ha?
- Qual è la relazione tra il tempo di esecuzione di *insertion sort* e il numero di inversioni nell'array di input? Spiegate la risposta
- Create un algoritmo che determina il numero di inversioni in una permutazione di n elementi nel tempo $\Theta(n\log n)$ nel caso peggiore (suggerimento: modificate *merge sort*)

Homework 1.2. Unimodal search

Un array è detto ***unimodale*** se è formato da una sequenza crescente seguita da una decrescente; più precisamente, se c'è un indice $m \in \{1, 2, \ldots, n\}$ tale che:

- $A[i] < A[i] + 1$, per ogni $1 \leq i < m$, *e*
- $A[i] > A[i] + 1$, per ogni $m \leq i < n$.

A[m] è l'elemento massimo, ed è l'unico elemento "massimo locale" circondato elementi più piccoli (A[m -1] e A[m+1]).

Si fornisca un algoritmo per calcolare il massimo elemento di un array unimodale A[1, … n] che esegue in $O(\lg n)$. Si dimostri che la complessità è $O(\lg n)$. (Suggerimento: esiste una soluzione molto simile alla ricerca binaria)

Homework 1.3. Ricorrenze

- Fornire il limite inferiore e superiore per $T(n)$ nelle seguente ricorrenza (si assume che T(n) è costante per $n \leq 10$):

  $T(n) = T(n/2) + 2^n$