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 #2

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 2.1. 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$n$)).

- $T(n)=T(\sqrt{n}) + \Theta(\log \log n)$ (suggerimento: si operi una sostituzione di variabili)
- $T(n)=10T(n/3)+17n^{1.2}$

Esercizio 2.2. Ordinare n numeri compresi nell'intervallo da 0 a n^3-1 nel tempo O(n). (Si utilizzi *radix sort* e *counting sort*).

Homeworks di verifica

Homework 6.1. Costruire un heap mediante inserimento
Possiamo costruire un *heap* chiamando ripetutamente la procedura MAX-HEAP-INSERT per inserire gli elementi nell'*heap*. Considerate la seguente variante della procedura BUILD-MAX-HEAP.

```
BUILD-MAX-HEAP_v2(A)
A.heap_size = 1
for (i=2) to A.length
        MAX-HEAP-INSERT(A, A[i])
```

a) Le procedure BUILD-MAX-HEAP e BUILD-MAX-HEAP_v2 creano sempre lo stesso *heap* se vengono eseguite con lo stesso array di input? Dimostrare che lo fanno o fornite un controesempio.
b) Dimostrare che, nel caso peggiore, BUILD-MAX-HEAP_v2 richiedo un tempo $\Theta(n\log n)$ per costruire un heap di n elementi.

Homework 6.2. Sorting almost sorted list (MIT 2)

Harry Potter, il famoso bambino mago di Hogwarts, è ancora una volta nei guai. Il professor Piton ha mandato Harry in prigione e gli ha assegnato il compito di sistemare tutti i vecchi compiti a casa degli ultimi 200 anni. Essendo un mago, Harry agita la sua bacchetta e dice, *ordinatus sortitus*, ed i fogli si impilano rapidamente in ordine.

All'uscita da prigione, Harry incontra la sua amica Hermione. È sconvolto perché il professor Piton ha scoperto che il suo incantesimo ha fallito. Invece di ordinare i fogli correttamente, ogni carta si trovava entro *k slot* dalla posizione corretta. Hermione suggerisce immediatamente che l'*insertion sort* avrebbe facilmente risolto il problema.

In questo problema, mostriamo che Hermione ha ragione. Sia A[1 . . n] un array di *n* elementi distinti:

(a) In primo luogo, definiamo una "**inversione**": se i<j e A[i] > A[j], allora la coppia (i,j) è detta inversione di A. Quale permutazione dell'array {1, 2, . . . , n} ha il maggior numero di inversioni? Quanti ne ha?

(b) Dimostrare che, se ogni foglio è inizialmente entro *k* slot dalla sua posizione corretta, l'*insertion sort* viene eseguito nel tempo O(*nk*). *Suggerimento: in primo luogo, si mostri che INSERTION-SORT(A) viene eseguito nel tempo O(n + I), dove I è il numero di inversioni in A.*

(c) SI progetti un algoritmo che soddisfa il limite inferiore, cioè, ordina una lista in cui ogni foglio si trova entro k slot dalla sua posizione corretta in tempo (n lg k). Suggerimento: *Si usi un Heap*