Informatica Generale
Primo Esonero

Giancarlo Bongiovanni, Tiziana Calamoneri, Ivano Salvo
Sapienza Università di Roma

16 Aprile 2018

Esercizio 1 (**8 punti**) Si consideri la seguente funzione in pseudocodice, che viene richiamata su un vettore bidimensionale $M[1..n][1..n]$ di interi, e che calcola il massimo valore contenuto in M:

```
funzione Max-Ricorsivo (M: vettore; n: interi)
    k ← M[n][n];
    for i = 1 to n do k ← max(k, M[n, i]);
    for i = 2 to n do k ← max(k, M[i, n]);
    if (n = 1) return k
    return max(k, Max-Ricorsivo (M, n − 1)).
```

Da essa si ricavi l'equazione di ricorrenza che ne esprime il costo computazione (**1.5 punti**). Inoltre, si risolva l'equazione di ricorrenza trovata utilizzando:

a. (**2 punti**) il metodo iterativo;
b. (**2 punti**) il metodo dell'albero;
c. (**2.5 punto**) il metodo di sostituzione.

Esercizio 2 (**12 punti**) Considerare il problema di trovare il *minimo* intero *non presente* in un vettore di interi non negativi *distinti*. Ad esempio, nel vettore: $\{3, 8, 4, 5, 11, 15, 9, 2, 6, 0, 1\}$ il minimo intero non presente è il 7.

1. (**4 punti**) Scrivere una funzione C di prototipo `int minFree(int v[], int n)` che restituisce il minimo intero non presente in `v`. La funzione `minFree` *non deve* modificare il vettore `v` e *non può* allocare strutture dati di dimensione dipendente da n. Valutare, anche informalmente, la complessità della funzione.

2. (**5 punti**) Osservando che il minimo intero non presente in `v` *deve* necessariamente essere un numero nell'intervallo $[0, n]$, dove n è la lunghezza del vettore, fornire una funzione `int minFreeLin(int v[], int n)` di complessità lineare in n che risolve lo stesso problema avendo la libertà di allocare un vettore di opportuna lunghezza locale alla funzione `minFree`.

3. (**1 punto**) Non potendo allocare memoria come al punto 1, ma potendo modificare il vettore, in quale altro modo è possibile accelerare la funzione scritta al punto 1? [non è richiesto codice per questo punto]

4. (**2 punti** Facoltativo, Impegnativo) Adattando opportunamente l'algoritmo *quickSort* (in particolare la funzione *partiziona*), scrivere una funzione C basata su *divide et impera* ricorsiva di complessità lineare che non alloca memoria, ma modifica `v`, senza necessariamente riordinarlo tutto.

Suggerimento: scrivere una funzione ausiliaria ricorsiva di prototipo `int minFreeAux(int v[], int inf, int sup)` che viene chiamata rispettando la precondizione che il minimo numero libero sia nell'intervallo $[inf, sup]$.

Esercizio 3 (**12 punti**) Si consideri un vettore $A[1..n]$ di n numeri interi positivi.

Si scrivano due algoritmi, uno iterativo ed uno ricorsivo, che dato A, stampino il massimo ed il minimo dei valori contenuti in A.

Di tali algoritmi:

a. (**3 punti**) si dia la spiegazione a parole (evidenziando le eventuali differenze tra la versione iterativa e quella ricorsiva);

b. (**2+3 punti**) si scrivano i due pseudocodici;

c. (**1.5+2.5 punti**) si calcolino i costi computazionali.