Informatica Generale
Primo Esonero

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

10 aprile 2017

Esercizio 1 (**8 punti**) Si consideri la seguente funzione in pseudocodice, che viene richiamata su un vettore V di interi e sui valori rispettivamente 1 ed n di `primo` e `ultimo`:

```
funzione MagicFunction (V:  vettore; primo, ultimo:  intero)
   if (primo >= ultimo) return
   for i = primo to ultimo
        stampa (V[i]);
   offset ← FLOOR ((ultimo-primo+1)/4);
   MagicFunction(V, primo, primo+offset-1);
   MagicFunction(V, ultimo-offset+1, ultimo);
return
```

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. (**3 punti**) il metodo iterativo;
b. (**2 punti**) il metodo dell'albero;
c. (**1.5 punto**) il metodo principale.

In tutti i casi è necessario dettagliare il procedimento usato e giustificare le proprie risposte.

Esercizio 2 (**10 punti**) Considerare il problema di trovare il sotto-vettore di somma massima in un vettore di numeri. Il problema è interessante solo se il vettore contiene sia numeri positivi che negativi (se contenesse solo positivi, il sotto-vettore di somma massima è il vettore stesso, mentre se contenesse solo negativi, sarebbe il vettore vuoto). Ad esempio, dato il vettore in Fig. 1, il sotto-vettore di somma massima è quello compreso tra gli indici 2 e 6 (compresi).

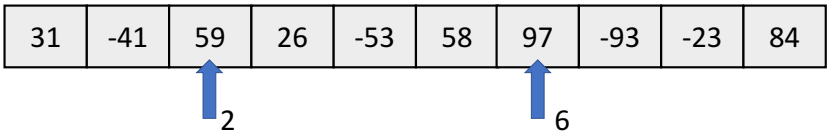

Figura 1: Esempio di vettore e relativo sotto-vettore di somma massima.

1. (**6 punti**) Supponete di avere a disposizione un esecutore C la cui unica capacità aritmetica sia una funzione `sumV(int a[], int inf, int sup)` che calcola la somma del vettore tra gli indici *inf* e *sup* (più precisamente calcola $\sum_{i=inf}^{sup-1} a[i]$). Avete ovviamente a disposizione gli usuali operatori di confronto (`<`, `<=` etc.). Scrivere una funzione `int svSommaMax(int a[], int n, int* b, int* e)` che dato il vettore a di interi di lunghezza n, restituisce la somma del sotto-vettore di somma massima e carica nelle variabili `b` ed `e` gli indici di dove comincia e finisce tale sotto-vettore (seguendo il galateo del C, nel nostro esempio, si dovrebbe tornare 187 e caricare rispettivamente 2 e 7 in `b` ed `e`).

Supponendo `sumV` abbia complessità $sup - inf$, dare la complessità della funzione `svSommaMax`, giustificandola (anche con un argomento informale).

2. (**4 punti**) Avendo a disposizione l'usuale esecutore C (quindi potendo liberamente sommare a vostro piacimento gli elementi del vettore) scrivere una funzione asintoticamente più efficiente. È sufficiente un algoritmo $\mathcal{O}(n^2)$, che usa tecniche standard (riflettere su come sia possibile calcolare incrementalmente le somme parziali). Se siete amanti del *divide-et-impera*, potete cimentarvi in un sofisticato algoritmo stile *mergeSort* di complessità $\mathcal{O}(n \log n)$. Se, infine, avete un po' di talento visionario per la semplicità, potreste scoprire un semplicissimo (ma non ovvio da trovare) algoritmo lineare.

Esercizio 3 (**12 punti**) Si consideri un vettore $A[1..n]$ di n numeri reali distinti e un intero $k \in \{1, \ldots, n\}$; il vettore è non ordinato.

Si scrivano due algoritmi, uno iterativo ed uno ricorsivo, che dati A e k, stampino i k valori più grandi presenti in A. Il costo degli algoritmi deve essere al piú $\mathcal{O}(kn)$.

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.