Informatica Generale
Primo Esonero

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

22 aprile 2016

Esercizio 1 (10 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
        for j = primo to ultimo
             stampa (V[i] + V[j]);
   medio → FLOOR ((primo+ultimo)/2);
   MagicFunction(V, primo, medio);
   MagicFunction(V, medio+1, ultimo);
return
```

Da essa si ricavi l'equazione di ricorrenza che ne esprime il costo computazione (**1 punto**). Inoltre, si risolva l' equazione di ricorrenza trovata utilizzando:

a. (**3 punti**) il metodo iterativo;
b. (**2 punti**) il metodo dell'albero;
c. (**1 punto**) il metodo principale;
d. (**3 punti**) il metodo di sostituzione.

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

Esercizio 2 (12 punti) Considerare le combinazioni di n oggetti presi k a k. Supponiamo di rappresentare una di tali combinazioni con un vettore di k interi compresi tra 1 ed n memorizzati nel vettore in ordine *crescente*.

1. **(6 punti)** Scrivere una funzione C `int nextCbn(int c[], int n, int k)` che presa in input una combinazione c di k interi tra 1 ed n, modifica c

in modo che rappresenti la prossima combinazione nell'ordine lessicografico e torna 1. Se la combinazione c in ingresso è l'ultima, `nextCbn` torna 0.

Ad esempio, se $n = 90$, $k = 6$ e c è il vettore $\{85, 86, 87, 88, 89, 90\}$ `nextCbn` torna 0. Se c è $\{1, 2, 3, 88, 89, 90\}$, `nextCbn` torna 1 e modifica c in modo che rappresenti la combinazione seguente nell'ordine lessicografico, cioè $\{1, 2, 4, 5, 6, 7\}$.

2. **(3 punti)** Usare la funzione `nextCbn` del punto precedente per scrivere una funzione iterativa C `allCbn(int n, int k)` che stampa in ordine crescente *tutte* le combinazioni di interi da 1 a n presi k a k [OSSERVAZIONE: non occorre memorizzare le combinazioni!].

3. **(3 punti)** Scrivere una funzione C ricorsiva `allCbnRec(int n, int k)` che usa lo schema ricorsivo del programma che calcola i coefficienti binomiali (sotto) per stampare tutte le combinazioni di interi da 1 a n presi k a k,

```
int cbin(int n, int k){
    if (k==0 || n==k) return 1;
return cbin(n-1,k-1)+cbin(n,k);
}
```

[SUGG: scrivere una funzione ausiliaria con parametri aggiuntivi, di cui uno un vettore in cui memorizzare la combinazione in costruzione. La prima chiamata ricorsiva genera le combinazioni che contengono un certo numero (con ancora $k-1$ elementi da posizionare), mentre la seconda genera le combinazioni che *non* lo contengono (con ancora k elementi da posizionare)]

Esercizio 3 (10 punti) Un min-heap è una struttura dati astratta rappresentata da un albero binario completo o quasi completo i cui nodi contengono un'informazione; per ogni nodo deve risultare che l'informazione memorizzata in esso è non maggiore delle informazioni memorizzate sui suoi figli.

Sia dato un vettore $A[1..n]$ contenente n numeri interi distinti.

Si progettino due algoritmi, uno iterativo ed uno ricorsivo (il più efficienti possibile) che, preso in input A, restituisca TRUE se A rappresenta un min-heap, e FALSE altrimenti.

Di tali algoritmi:

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