Algoritmi e Strutture Dati con Laboratorio (modulo I)

Testo della prova scritta del 16 luglio 2013

docente: Luciano Gualà

Cognome:..... Nome:..... Matr.:..... Corso di Laurea:.....

Esercizio 1 [10 punti]

(a) Si ordinino le seguenti funzioni in ordine non decrescente di tasso di crescita asintotica. Per ogni coppia di funzioni $f_i(n), f_{i+1}(n)$ adiacenti nell'ordinamento si specifichi se $f_i(n) = \Theta(f_{i+1}(n))$ o se $f_i(n) = o(f_{i+1}(n))$.

Le funzioni sono: 3^n, $\frac{n^2\sqrt{n^3+1}}{\sqrt{n+9}}$, $n^2\sqrt{n} + n^3$, 2^n, $\frac{n^3}{\log n^{33}}$, $\frac{n^3+313}{\log\log n}$, $n^{\log\log n}$, $\frac{7n^{13}-\log n}{3}$, $n^{2.999}$.

(b) Per un problema sono noti due algoritmi ricorsivi, A_1 e A_2 le cui complessità temporali sono descritte dalle seguenti equazioni di ricorrenza:

$T_1(n) = T_1(n-1) + T(n-2) + 1$, $T_1(1) = 1$;

$T_2(n) = 4\,T_2(n/16) + \sqrt{n} + \sqrt[3]{\log n}$, $T_2(1) = 1$;

Dire, motivando la risposta, quale algoritmo è preferibile usare.

Esercizio 2 [12 punti] Sia M una matrice $k \times n$ di valori distinti. I valori in ognuna delle k righe sono ordinati in modo crescente. Progettare un algoritmo che, presa M, restituisca il valore *mediano* della matrice, ovvero il valore che, se si considerano in ordine tutti gli elementi della matrice dal più piccolo al più grande si trova in posizione $\lfloor \frac{nk}{2} \rfloor + 1$. L'algoritmo deve avere complessità temporale $O(nk \log k)$ e deve usare memoria ausiliaria $O(k)$.

Esercizio 3 [13 punti] Homer Simpson sta facendo una sfida in cui deve mangiare più donuts (morbide ciambelline fritte ricoperte di glassa) possibile. La sfida funziona in n round. Al round i vangono servite a Homer x_i donuts. La quantità di donuts che Homer riesce a mangiare in un certo round i dipende da quanto tempo non ha mangiato e cresce esponenzialmente con la fame. Per essere più precisi, se al round i Homer ha passato (cioè non ha mangiato ne)i precedenti j round, lui riuscirà a mangiare $\min\{2^j, x_i\}$ donuts, e la sua fame si azzerà, così che nel prossimo turno ne potrà mangiare una sola, ma se saltasse il turno successivo a quello dopo ne potrebbe mangiare 2 e così via. Assumeremo che Homer arrivi alla sfida con la pancia piena e che quindi al primo turno la sua fame gli permette di mangiare $2^0 = 1$ sola donut.

Progettare un algoritmo di programmazione dinamica che calcoli il numero massimo di donuts che Homer riesce a mangiare.

Di seguito è mostrato un esempio con $n = 4$. La strategia ottima, in questo caso, è mangiare al terzo e al quarto round.

round i:	1	2	3	4
x_i	1	10	10	1
fame:	1	2	4	8