

Introduzione agli Algoritmi
Prova intermedia - 14 Aprile 2014
Prof. Emanuela Fachini (canale 1) e Prof. Irene Finocchi (canale 2)

Le risposte non motivate non saranno prese in considerazione.

Negli esercizi di progettazione, prima di passare allo pseudocodice descrivete l'idea algoritmica sottostante. Per tutti gli algoritmi progettati è necessario analizzare tempo di esecuzione e correttezza.

Testo 3

Esercizio 1

Si consideri la seguente funzione:

```
test (intero n)
  if n ≤ 36 then return 1
  k = 1
  while k ≤ (n*n) do k = k*4
  return k + test(n/6)
```

Scrivere la relazione di ricorrenza che descrive il tempo di esecuzione $T(n)$ della funzione *test* e dimostrare che $T(n) = O(\log^2 n)$.

Esercizio 2

Definiamo il problema del *salto rispetto all'indice* come segue:

data una sequenza ordinata di n interi positivi distinti, $a_1 < a_2 < \dots < a_n$, determinare – se esiste – il più piccolo indice i tale che $a_i > i$.

Progettare un algoritmo che risolva il problema del salto rispetto all'indice in tempo $O(\log n)$. Analizzare sia la correttezza dell'algoritmo proposto che il tempo di esecuzione nel caso peggiore. Si osservi che in una sequenza come quella descritta non può mai accadere che $a_i < i$.

Esercizio 3

Quali delle seguenti affermazioni sono vere? Motivare la risposta con una dimostrazione o con un opportuno esempio.

Introduzione agli Algoritmi
Prova intermedia - 14 Aprile 2014
Prof. Emanuela Fachini (canale 1) e Prof. Irene Finocchi (canale 2)

1. Siano $f(n)$, $g(n)$, $t(n)$ e $s(n)$ quattro funzioni tali che: $f(n)=O(g(n))$ e $t(n)=O(s(n))$. Allora $f(n)+t(n) = O(\max\{g(n), s(n)\})$
2. Siano $f(n)$, $t(n)$ e $s(n)$ tre funzioni tali che: $f(n)=O(t(n))$ e $t(n)=\Omega(s(n))$. Allora $f(n) = \Omega(s(n))$.
3. Sia $f(n)$ il tempo di esecuzione nel caso migliore di un algoritmo che risolve un problema \mathcal{P} . Se \mathcal{P} ha una complessità $\Omega(g(n))$, allora può accadere che $f(n)=O(g(n))$ ma $f(n)\neq\Theta(g(n))$.
4. Sia $f(n)$ il tempo di esecuzione nel caso peggiore di un algoritmo che risolve un problema \mathcal{P} . Se \mathcal{P} ha una complessità $\Omega(g(n))$, allora deve essere $f(n)=\Omega(g(n))$