Introduzione agli Algoritmi
Appello del 4 Giugno 2013
Prof. Emanuela Fachini e Prof. Irene Finocchi

Parte 1 (t*empo concesso: 1h 30m)*

Le risposte non motivate non saranno prese in considerazione. Negli esercizi di progettazione, prima di scrivere lo pseudocodice occorre illustrare l'idea algoritmica soggiacente, specificando in dettaglio le funzioni che si utilizzano. ***Non saranno corrette soluzioni*** *in puro pseudocodice o che utilizzino funzioni il cui comportamento non è stato specificato, anche se introdotte durante il corso.*

Per tutti gli algoritmi progettati è inoltre necessario analizzare non solo il tempo di esecuzione, ma anche la correttezza (per il primo canale: per ogni ciclo introdotto si definisca un'invariante e la si usi per dimostrare la correttezza).

Esercizio 1

Un *puzzle lineare* è una sequenza di tessere in cui ognuna di esse (tranne la prima e l'ultima) si incastra sia con la tessera che la precede (quella "a sinistra") che con quella che la segue (quella "a destra"). Naturalmente la prima tessera si incastra solo con quella che la segue, e l'ultima solo con quella che la precede.

Assumete che:

- ogni tessera è rappresentata da due numeri, il primo ≤ 0 e il secondo ≥ 0;
- un numero positivo si incastra solo con il suo opposto;
- i bordi del puzzle (sia il sinistro che il destro) sono rappresentati dal numero 0;
- in ogni sequenza di tessere un numero ed il suo opposto compiano al massimo una volta

Un esempio di puzzle è il seguente: <0, 12> <-12, 4> <-4, 54> <-54, 0>

1. Progettate un algoritmo che, data una sequenza arbitraria di *n* tessere, ricompone il puzzle (o restituisce FALSE se ciò è impossibile). Immaginate che le tessere a disposizione siano memorizzate in un array A di *n* coppie, originariamente disordinato, dove A[i].sx e A[i].dx rappresentano rispettivamente il bordo sinistro e il bordo destro della i-esima tessera, con i $\in$ [1,n].

2. Discutete il tempo di esecuzione dell'algoritmo proposto, che dovrebbe essere *O(n log n),* e la correttezza (per il primo canale utilizzando gli invarianti nel caso di cicli).

Esercizio 2

Basandovi sulla definizione di O, Ω e Θ, rispondete alle seguenti domande motivando la risposta:

1. E' possibile che un algoritmo abbia tempo di esecuzione $O(n^2)$ nel caso peggiore e $\Omega(n)$ nel caso migliore?

2. Se un algoritmo ha tempo di esecuzione $\Omega(n^2)$ nel caso peggiore, è possibile che nel caso migliore l'algoritmo abbia complessità O(n log n)?

3. Se un algoritmo ha tempo di esecuzione $\Omega(n)$ e $O(n)$ nel caso peggiore, possiamo concludere che nel caso peggiore è $\Theta(n)$?

4. Se la complessità di un *problema* è $\Omega(n^2)$ nel caso peggiore, e si dispone di un algoritmo che risolve il problema in tempo $O(n^2)$ nel caso peggiore, si può affermare che l'algoritmo ha tempo di esecuzione $\Theta(n^2)$ nel caso peggiore?

Introduzione agli Algoritmi
Appello del 4 Giugno 2013
Prof. Emanuela Fachini e Prof. Irene Finocchi

Parte 2 *(tempo concesso: 1h 30m)*

Le risposte non motivate non saranno prese in considerazione. Negli esercizi di progettazione, prima di scrivere lo pseudocodice occorre illustrare l'idea algoritmica soggiacente, specificando in dettaglio le funzioni che si utilizzano. ***Non saranno corrette soluzioni*** *in puro pseudocodice o che utilizzino funzioni il cui comportamento non è stato specificato, anche se introdotte durante il corso.*

Per tutti gli algoritmi progettati è inoltre necessario analizzare non solo il tempo di esecuzione, ma anche la correttezza (per il primo canale: per ogni ciclo introdotto si definisca un'invariante e la si usi per dimostrare la correttezza).

Esercizio 3

1. Qual è l'albero binario che - a parità di altezza *h* - ha il massimo numero *n* di nodi? Qual è questo massimo numero di nodi in funzione dell'altezza? Qual è quindi la minima altezza di un albero binario in termini del numero *n* dei nodi?

2. Qual è l'albero binario che - a parità di altezza *h* - ha il minimo numero *n* di nodi? Qual è questo minimo numero di nodi in funzione dell'altezza? Qual è quindi la massima altezza di un albero binario in termini del numero *n* dei nodi?

3. Sotto quali ipotesi la rappresentazione in memoria di un albero binario tramite *array posizionale* è conveniente dal punto di vista dell'uso dello spazio di memoria ? In quali casi invece non lo è?

Esercizio 4

Dimostrare che ogni algoritmo generale di ordinamento basato su confronti esegue nel caso peggiore $\Omega(n \log n)$ confronti.

Esercizio 5

Progettare un algoritmo che prenda in input un albero binario T ed un intero *k* e restituisca TRUE se esiste un cammino *radice-foglia* la cui somma delle chiavi è uguale a *k*.

Dimostrare la correttezza ed analizzare il tempo di esecuzione dell'algoritmo proposto, assumendo che T sia rappresentato tramite puntatori al figlio sinistro e al figlio destro.