Introduzione agli Algoritmi
Appello del 23 Gennaio 2014
Proff. Emanuela Fachini ed Irene Finocchi

Parte 1

Le risposte non motivate non saranno prese in considerazione.
Negli esercizi di progettazione, prima di passare allo pseudocodice descrivete l'idea algoritmica soggiacente.
Per tutti gli algoritmi progettati è necessario analizzare tempo di esecuzione e correttezza (canale 1: usare gli invarianti per progettare e verificare il ciclo in caso di algoritmi iterativi).

Esercizio 1
Si considerino le seguenti funzioni:

```
fun (intero n)
    i=1
    while i≤n do
        test(i)
        i = i*2
```

```
test (intero x)
    if x≤1 then return 1
    k = 0
    for i ← 1 to x do k++
    test(k/4)
```

Scrivere la relazione di ricorrenza che descrive il tempo di esecuzione T(x) della funzione *test* e risolverla. Calcolare poi il tempo di esecuzione F(n) della funzione *fun*.

Esercizio 2
Una matrice A di *n* x *m* interi si dice *monotona* se ogni riga e ogni colonna è ordinata in ordine crescente. Supponiamo che gli elementi di A siano tutti diversi.
Esempio di matrice 4x5 monotona:

2	**5**	**8**	**10**	**11**
3	**7**	**9**	**12**	**13**
6	**8**	**10**	**15**	**16**
10	**20**	**30**	**40**	**41**

Si descriva un algoritmo che, data una matrice A monotona, trovi un elemento *k* nella matrice A in *O(n+m)* passi, restituendo gli indici di riga *i* e di colonna *j* tali che *A[i][j]=k*.

Si dimostri la correttezza dell'algoritmo proposto e si motivi il risultato sul tempo di esecuzione.

Introduzione agli Algoritmi
Appello del 23 Gennaio 2014
Proff. Emanuela Fachini ed Irene Finocchi

Le risposte non motivate non saranno prese in considerazione.
Negli esercizi di progettazione, prima di passare allo pseudocodice descrivete l'idea algoritmica soggiacente.
Per tutti gli algoritmi progettati è necessario analizzare tempo di esecuzione e correttezza (canale 1: usare gli invarianti per progettare e verificare il ciclo in caso di algoritmi iterativi).

Esercizio 3
Rispondere alle seguenti domande:

1. E' possibile che un ABR T, completo almeno fino al penultimo livello, contenente almeno tre nodi ed avente chiavi distinte soddisfi anche le proprietà di ordinamento tra padri e figli degli heap binari? Se si, mostrare un esempio di tale albero. Se no, dimostrarne l'impossibilità.
2. E' possibile che un ABR T contenente almeno tre nodi ed avente chiavi distinte soddisfi anche le proprietà di ordinamento tra padri e figli degli heap binari? Se si, mostrare un esempio di tale albero. Se no, dimostrarne l'impossibilità.

Esercizio 4
Sia T un ABR nel quale *due chiavi sono state scambiate*. Si scriva un algoritmo che trovi le due chiavi fuori posto e, scambiandole, ripristini la proprietà di ricerca in T.

- Si consideri prima il caso più semplice in cui lo scambio riguarda padre e figlio.
- Si consideri poi il caso più generale in cui lo scambio è avvenuto tra due chiavi in posizioni qualsiasi nell'albero.

Si dimostri la correttezza e si analizzi il tempo di esecuzione dell'algoritmo proposto (il tempo di esecuzione dovrebbe essere O(n), dove n è il numero di nodi di T).

Esempio: