

ESAME DI ALGORITMI E STRUTTURE DATI
A.A. 2014/2015 – APPELLO DEL 22/06/2015

TEMPO A DISPOSIZIONE: 120 MINUTI

ESERCIZIO 1:

Si consideri di avere a disposizione una struttura dati di tipo "Albero Binario" ottenuta utilizzando una rappresentazione collegata di tipo "puntatore ai figli".

Sia N un nodo dell'albero. Si supponga di avere a disposizione le seguenti funzioni:

$N.figlioSx() \rightarrow nodo$	Restituisce il nodo che è figlio sinistro del nodo N . Se N non ha un figlio sinistro la funzione restituisce NULL.
$N.figlioDx \rightarrow nodo$	Restituisce il nodo che è figlio destro del nodo N . Se N non ha un figlio destro la funzione restituisce NULL.
$N.colore() \rightarrow stringa$	Restituisce la stringa memorizzata nel nodo N . Tale stringa rappresenta il "colore" del nodo. Ogni nodo ha uno ed un solo colore. I colori possibili sono: "Marrone", "Giallo" e "Verde".

Scrivere la funzione $foglieVerdi(nodo\ N) \rightarrow Boolean$

la quale, preso in ingresso un albero binario radicato in N , restituisce *VERO* se tutte le foglie dell'albero sono "Verdi", *FALSO* altrimenti.

ESERCIZIO 2:

1) Scrivere l'algoritmo

$estMaxHeap(array\ A) \rightarrow Boolean$

il quale, preso in ingresso un array di interi A di dimensione n , termina restituendo *TRUE* se A rappresenta un **MAX-HEAP** e *FALSE* altrimenti.

2) Calcolare la complessità nel *caso peggiore* e nel *caso migliore* dell'algoritmo proposto, motivando la risposta.

ESERCIZIO 3:

Sia **P** una pila contenente i seguenti valori { 5, 4, 7, 1, 8, 2 } dove '5' è l'ultimo elemento inserito nella pila.

Fornire, motivando dettagliatamente la risposta, la sequenza di interi stampati dal seguente algoritmo quando come input viene fornita la pila **P**.

ATTENZIONE: la funzione `stampaVisitaSimmetrica(nodo n)` stampa i nodi dell'albero radicato nel nodo **n** nell'ordine in cui questi sono visitati da una *visita simmetrica*.

```
stampa(pila P)
  sia A un albero binario di ricerca;
  while(not P.isEmpty())
    A.insert(P.pop());
  stampaVisitaSimmetrica(A.root()); // la funzione A.root()
                                     // restituisce la radice
                                     // dell'albero A
```

ESERCIZIO 4:

- Si scriva lo pseudocodice dell'algoritmo `SelectionSort` in grado di ordinare gli elementi contenuti in un array **A** di **n** interi preso in ingresso.
- Descrivere i passi eseguiti dall'algoritmo `SelectionSort` sul seguente array in input.

0	1	2	3	4	5	6
4	2	5	9	3	8	6

- Indicare la complessità dell'algoritmo `SelectionSort` nel caso peggiore.

ESERCIZIO 5:

Fornire la rappresentazione con *lista di adiacenza* del grafo orientato rappresentato in figura.

