Algoritmi e Strutture di Dati – A.A. 2011-2012
Seconda Prova dell'appello del 12 e 13 giugno 2012

SOLUZIONI DELLA PSEUDOCODIFICA

Negli esercizi seguenti supponi che un grafo diretto sia rappresentato con un array A in cui ogni elemento A[*u*] è un riferimento al primo elemento della lista di adiacenza doppiamente concatenata del nodo *u*. Un albero binario T ha invece il solo riferimento alla radice T.root (ogni nodo dell'albero ha i campi parent, left, right e info).

Esercizio 1

Scrivi lo pseudocodice della procedura FIGLI-RADICE(T) che prende in input un albero T e restituisce **true** se la radice dell'albero ha due figli, **false** altrimenti (un solo figlio o nessun figlio).

```
FIGLI-RADICE(T)   /* assumo che l'albero abbia sempre la radice */
     return (T.root.left != NIL) AND (T.root.right != NIL)
```

oppure:

```
FIGLI-RADICE(T) /* T potrebbe essere anche un albero vuoto */
     if (T.root == NIL) return FALSE
     return (T.root.left != NIL) AND (T.root.right != NIL)
```

Esercizio 2

Scrivi lo pseudocodice della procedura CERCA(T,*u*) che prende in input l'albero T e un valore intero *u*, e restituisce un riferimento al nodo che ha come valore info il valore *u*.

```
CERCA(T, u) /* T = albero, u = indice di un nodo */
     return CERCA_RIC(T.root, u)

CERCA_RIC(n, u) /* n = riferimento ad un nodo (anche NIL), u = indice */
     if (n == NIL) return NIL
     if (n.info == u) return n
     sin = CERCA_RIC(n.left, u)
     if (sin != NIL) return sin
     else return CERCA_RIC(n.right,u)
```

Esercizio 3

Scrivi lo pseudocodice della procedura CAMMINO(A,*u*,*v*) che prende in input un grafo A e verifica se esite un cammino diretto dal nodo con indice *u* al nodo con indice *v*.

```
CAMMINO(A,u,v)
     /* B è un nuovo array di interi con A.length posizioni */
     /* B[i] = 1 se ho già lanciato una ricerca di v a partire da i */
     for i = 0 to B.length-1
          B[i] = 0        /* ancora non lanciata nessuna ricerca da i */
     return CAMMINO_RIC(A,B,u,v)
```

```
CAMMINO_RIC(A,B,i,v)
      B[i] = 1         /* lanciato CAMMINO_RIC a partire da i */
      x = A[i]
      while x != NIL
            key = x.info
            if (key == v) return TRUE
            if (B[key] != NIL) /* se ancora non ho cercato da key */
                  raggiunto = CAMMINO_RIC(A,B,key,v)
                  if (raggiunto) return TRUE
            x = x.next
      return FALSE
```

gli studenti 270 avrebbero potuto svolgere questo esercizio con una visita del grafo

```
CAMMINO(A,u,v)
      /* color è un nuovo array di interi con A.length posizioni */
      for i=0 to color.length-1
            color[i] = 0
      DFS(A,u,color)
      if (color[v] == 2) return TRUE
      else return FALSE

DFS(A,u,color) → vedi appunti delle lezioni
```

Esercizio 4 (solo studenti D.M. 270/04)

Scrivi lo pseudocodice della procedura ANTENATO(T,*u*,*v*) che prende in input un albero T e gli interi *u* e *v* e ritorna **true** se il nodo con etichetta *u* è un antenato del nodo con etichetta *v*, **false** altrimenti.

```
ANTENATO(T,u,v) /* assumo ogni nodo antenato di se stesso */
      n = CERCA(T,u)
      v = CERCA_RIC(n,v)
      return (v != NIL)
```

Esercizio 5

Discuti la complessità computazionale (nel solo caso peggiore) delle procedure che hai proposto per gli esercizi precedenti, utilizzando *n* per denotare il numero totale dei nodi dell'albero o del grafo.

FIGLI_RADICE(T) $\in$ O(1) [in realtà Theta(1)]

CERCA(T,u) $\in$ O(n) [in realtà Theta(n), perché nel caso peggiore bisogna guardare tutto l'albero]

CAMMINO(A,u,v) $\in$ O(n^2) perché nel caso peggiore si guardano tutti gli archi [anche questo Theta(n^2)]

ANTENATO(T,u,v) $\in$ O(n) [Theta(n)] due ricerche in sequenza che nel caso peggiore sono entrambi lineari