Corso di Informatica 1 (IN110 - Fondamenti) – Prof. Marco Liverani – a.a. 2009/2010

Esame scritto del 12 febbraio 2010

Si richiede di risolvere entrambi gli esercizi riportando una codifica in linguaggio C completa dei due programmi. Nel caso in cui non si riesca a completare entrambi gli esercizi si suggerisce di riportare almeno la codifica in C delle funzioni principali o una loro pseudo-codifica. È possibile consultare libri e appunti personali, ma non scambiare libri o appunti con altri studenti. I compiti che presenteranno evidenti ed anomale "similitudini" saranno annullati. La prova scritta ha una durata di tre ore, durante le quali non è consentito allontanarsi dall'aula, se non dopo aver consegnato il compito. Si richiede di riportare sul foglio del compito il proprio nominativo completo ed il numero di matricola o un codice identificativo personale equivalente.

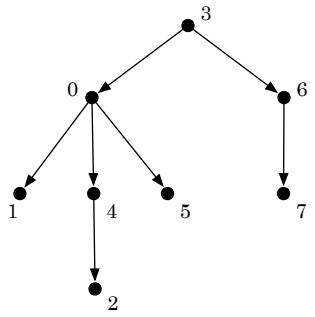
Esercizio n. 1

Leggere in input un vettore di n interi $\Pi = (\pi_0, \pi_1, \pi_2, \ldots, \pi_{n-1})$; il vettore rappresenta le relazioni padre-figlio tra i vertici di un albero T orientato con radice: se $\pi_i = j$ allora il vertice v_j dell'albero è il padre del vertice v_i; il vertice per cui risulta $\pi_k = -1$ è la radice e non ha alcun padre. Costruire le liste di adiacenza dell'albero T e stamparle.

Esempio Si consideri il seguente vettore

$$\Pi = (\pi_0 = 3, \pi_1 = 0, \pi_2 = 4, \pi_3 = -1, \pi_4 = 0, \pi_5 = 0, \pi_6 = 3, \pi_7 = 6)$$

L'albero di cui devono essere costruite le liste di adiacenza è rappresentato in figura:



Soluzione

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 100

struct nodo {
  int info;
  struct nodo *next;
};
```

```
10
11 int leggiVettore(int X[]) {
12   int n, i;
13   printf("Numero di elementi: ");
14   scanf("%d", &n);
15   printf("Inserire %d elementi: ", n);
16   for (i=0; i<n; i++)
17     scanf("%d", &X[i]);
18   return(n);
19 }
20
21 void stampaVettore(int X[], int n) {
22   int i;
23   for (i=0; i<n; i++)
24     printf("%d ", X[i]);
25   printf("\n");
26   return;
27 }
28
29 void costruisciAlbero(struct nodo *T[], int Pi[], int n) {
30   int i;
31   struct nodo *p;
32   for (i=0; i<n; i++)
33     T[i] = NULL;
34   for (i=0; i<n; i++) {
35     if (Pi[i] != -1) {
36       p = malloc(sizeof(struct nodo));
37       p->info = i;
38       p->next = T[Pi[i]];
39       T[Pi[i]] = p;
40     }
41   }
42   return;
43 }
44
45 void stampaLista(struct nodo *p) {
46   while (p != NULL) {
47     printf("%d --> ", p->info);
48     p = p->next;
49   }
50   printf("NULL\n");
51   return;
52 }
53
54 void stampaGrafo(struct nodo *G[], int n) {
55   int i;
56   for (i=0; i<n; i++) {
57     printf("%d: ", i);
58     stampaLista(G[i]);
59   }
60   return;
61 }
62
63
```

```
int main(void) {
  int n, Pi[MAX];
  struct nodo *T[MAX];
  n = leggiVettore(Pi);
  stampaVettore(Pi, n);
  costruisciAlbero(T, Pi, n);
  stampaGrafo(T, n);
  return(0);
}
```

Esercizio n. 2

Letto in input un numero $n > 0$, costruire la matrice A quadrata di ordine n, con i numeri interi $1, 2, \ldots, n(n+1)/2$ disposti secondo lo schema diagonale rappresentato nella seguente figura di esempio (in questo caso $n = 4$). Stampare gli elementi non nulli della matrice A.

$$A = \begin{pmatrix} 1 & 3 & 6 & 10 \\ 2 & 5 & 9 & 0 \\ 4 & 8 & 0 & 0 \\ 7 & 0 & 0 & 0 \end{pmatrix}$$

Soluzione

```
#include <stdlib.h>
#include <stdio.h>
#define MAX 100

void costruisciMatrice(int A[MAX][MAX], int n) {
  int i, j, c;
  c = 1;
  for (i=0; i<n; i++) {
    for (j=n-i; j<n; j++)
      A[i][j] = 0;
    for (j=0; j<=i; j++) {
      A[i-j][j] = c;
      c++;
    }
  }
  return;
}

void stampaMatrice(int A[MAX][MAX], int n) {
  int i, j;
  for (i=0; i<n; i++) {
    for (j=0; j<n && A[i][j]>0; j++)
      printf("%2d ", A[i][j]);
    printf("\n");
  }
  return;
}


```

```
30 int main(void) {
31   int n, A[MAX][MAX];
32   printf("Ordine della matrice: ");
33   scanf("%d", &n);
34   costruisciMatrice(A, n);
35   stampaMatrice(A, n);
36   return(0);
37 }
```