Corso di Informatica 1 (IN110 - Fondamenti) – Prof. Marco Liverani – a.a. 2012/2013

Esame scritto del 18 Gennaio 2013

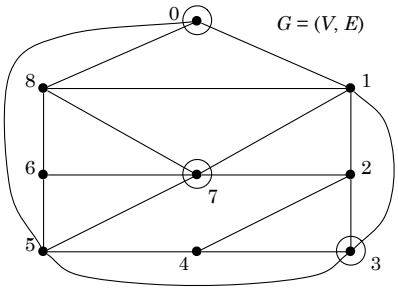
Si richiede di risolvere entrambi gli esercizi riportando una codifica in linguaggio C completa dei due programmi. Nel caso in cui non si riesca a completare entrambi gli esercizi si suggerisce di riportare almeno la codifica in C delle funzioni principali o una loro pseudo-codifica. È possibile consultare 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

Dato in input un grafo $G = (V, E)$, rappresentato con liste di adiacenza, e un vertice $u \in V$, stampare in output l'elenco dei vertici "quasi amici" di u.
Un vertice $v \in V$ è "quasi amico" di u se:

1. $v \neq u$;
2. v non è adiacente a u;
3. v e u hanno in comune almeno tre vertici adiacenti.

Esempio Si consideri il grafo G rappresentato in figura. Sia $u = 7$; i vertici "quasi amici" di u sono i vertici 0 e 3.



Soluzione

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 50

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

struct nodo *leggiLista(void) {
```

```
12    struct nodo *p, *primo = NULL;
13    int i, n;
14    printf("Numero di elementi: ");
15    scanf("%d", &n);
16    printf("Inserisci %d elementi: ", n);
17    for (i=0; i<n; i++) {
18      p = malloc(sizeof(struct nodo));
19      scanf("%d", &p->info);
20      p->next = primo;
21      primo = p;
22    }
23    return(primo);
24  }
25
26  int leggiGrafo(struct nodo *G[]) {
27    int i, n;
28    printf("Numero di vertici del grafo: ");
29    scanf("%d", &n);
30    for (i=0; i<n; i++) {
31      printf("Inserisci la lista dei vertici adiacenti a %d:\n", i);
32      G[i] = leggiLista();
33    }
34    return(n);
35  }
36
37  int quasiAmici(struct nodo *G[], int u, int v) {
38    struct nodo *p, *q;
39    int cont, rc=0;
40    if (u != v) {
41      p = G[u];
42      while (p != NULL && p->info != v)
43        p = p->next;
44      if (p == NULL) {
45        p = G[u];
46        cont = 0;
47        while (p != NULL && cont < 3) {
48          q = G[v];
49          while (q != NULL && q->info != p->info)
50            q = q->next;
51          if (q != NULL)
52            cont++;
53          p = p->next;
54        }
55        if (cont == 3)
56          rc = 1;
57      }
58    }
59    return(rc);
60  }
61
62  int main(void) {
63    struct nodo *G[MAX];
64    int n, u, v;
65    n = leggiGrafo(G);
```

```
  printf("Inserisci un vertice del grafo: ");
  scanf("%d", &u);
  printf("I vertici quasi amici di %d sono i seguenti: ", u);
  for (v=0; v<n; v++)
    if (quasiAmici(G, u, v))
      printf("%d ", v);
  printf("\n");
  return(0);
}
```

Esercizio n. 2

Letti in input due interi positivi $n, m < 100$ costruire una matrice M di ordine $n \times m$ composta da numeri interi casuali compresi tra 50 e 75 ($50 \leq M_{i,j} \leq 75$ per $i = 0, \ldots, n-1$, $j = 0, \ldots, m-1$). Stampare la matrice. Ordinare in ordine crescente gli elementi delle colonne di M. Stampare la matrice risultante.

Esempio Sia $n = 4$ e $m = 5$; di seguito riportiamo una matrice $n \times m$ di numeri casuali generata dal programma e la corrispondente matrice con le colonne ordinate:

$$A = \begin{pmatrix} 57 & 62 & 54 & 51 & 60 \\ 74 & 56 & 68 & 74 & 51 \\ 55 & 70 & 58 & 53 & 66 \\ 68 & 56 & 72 & 69 & 70 \end{pmatrix} \qquad A' = \begin{pmatrix} 55 & 56 & 54 & 51 & 51 \\ 57 & 56 & 58 & 53 & 60 \\ 68 & 62 & 68 & 69 & 66 \\ 74 & 70 & 72 & 74 & 70 \end{pmatrix}$$

Soluzione

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

void scambia(int *a, int *b) {
  int c;
  c = *a;
  *a = *b;
  *b = c;
  return;
}

void generaMatrice(int A[MAX][MAX], int n, int m) {
  int i, j;
  srand((unsigned)time(NULL));
  for (i=0; i<n; i++)
    for (j=0; j<m; j++)
      A[i][j] = rand() % 26 + 50;
  return;
}

void stampaMatrice(int A[MAX][MAX], int n, int m) {
  int i, j;
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++)
```

```
27      printf("%3d", A[i][j]);
28    printf("\n");
29  }
30  printf("\n");
31  return;
32 }
33
34 void ordinaMatrice(int A[MAX][MAX], int n, int m) {
35  int i, j, k, min;
36  for (j=0; j<m; j++) {
37    for (i=0; i<n-1; i++) {
38      min = i;
39      for (k=i+1; k<n; k++)
40        if (A[k][j] < A[min][j])
41          min = k;
42      scambia(&A[i][j], &A[min][j]);
43    }
44  }
45  return;
46 }
47
48 int main(void) {
49  int A[MAX][MAX], n, m;
50  printf("Inserisci il numero di righe e di colonne: ");
51  scanf("%d %d", &n, &m);
52  generaMatrice(A, n, m);
53  stampaMatrice(A, n, m);
54  ordinaMatrice(A, n, m);
55  stampaMatrice(A, n, m);
56  return(0);
57 }
```