Corso di Informatica 1 (IN110 - Fondamenti) – Prof. Marco Liverani – a.a. 2010/2011

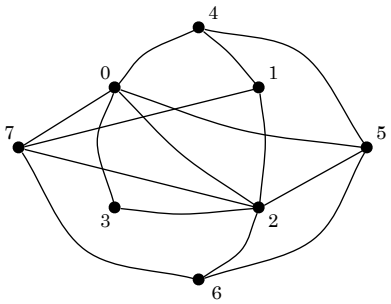
Esame scritto del 8 Luglio 2011

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

Le relazioni di "amicizia" fra gli utenti di un noto *social network* sono rappresentate mediante un grafo non orientato $G = (V, E)$, i cui vertici rappresentano gli utenti della rete e gli spigoli le relazioni di amicizia che legano alcune coppie di utenti. Letto in input il grafo G rappresentarlo mediante liste di adiacenza. Scelti tre utenti $a, b, c \in V(G)$ stampare l'elenco (senza ripetizioni) degli amici in comune a tutti e tre e l'elenco degli amici di c che non siano anche amici di a.

Esempio Si consideri il grafo G rappresentato in figura. Siano $a = 0, b = 1, c = 2$ i tre vertici scelti arbitrariamente su G. L'insieme degli amici in comune è $\{7\}$, mentre l'insieme degli amici di c che non siano amici di a è $\{1, 6\}$.



Soluzione

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

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

struct nodo *leggiLista(void) {
  struct nodo *p, *primo = NULL;
  int i, n;
```

```
13    printf("Numero di elementi: ");
14    scanf("%d", &n);
15    printf("Inserisci %d vertici: ", n);
16    for (i=0; i<n; i++) {
17      p = malloc(sizeof(struct nodo));
18      scanf("%d", &p->info);
19      p->next = primo;
20      primo = p;
21    }
22    return(primo);
23  }
24
25  int leggiGrafo(struct nodo *G[]) {
26    int i, n;
27    printf("Numero di vertici: ");
28    scanf("%d", &n);
29    for (i=0; i<n; i++) {
30      printf("Lista di adiacenza di %d\n", i);
31      G[i] = leggiLista();
32    }
33    return(n);
34  }
35
36  void stampaLista(struct nodo *p) {
37    while (p != NULL) {
38      printf("%d --> ", p->info);
39      p = p->next;
40    }
41    printf("NULL\n");
42    return;
43  }
44
45  void stampaGrafo(struct nodo *G[], int n) {
46    int i;
47    for (i=0; i<n; i++) {
48      printf("G[%d]: ", i);
49      stampaLista(G[i]);
50    }
51    return;
52  }
53
54  int adiacente(struct nodo *G[], int u, int v) {
55    struct nodo *p;
56    int r;
57    p = G[u];
58    while (p != NULL && p->info != v)
59      p = p->next;
60    if (p == NULL)
61      r = 0;
62    else
63      r = 1;
64    return(r);
65  }
66
```

```c
void stampaAmiciComuni(struct nodo *G[], int n, int a, int b, int c) {
  struct nodo *p;
  printf("Amici comuni a %d, %d e %d: ", a, b, c);
  p = G[a];
  while (p != NULL) {
    if (adiacente(G, b, p->info) && adiacente(G, c, p->info))
      printf("%d ", p->info);
    p = p->next;
  }
  printf("\n");
  return;
}

void stampaAmiciNonComuni(struct nodo *G[], int n, int a, int c) {
  struct nodo *p;
  printf("Amici di %d che %d non conosce: ", c, a);
  p = G[c];
  while (p != NULL) {
    if (!adiacente(G, a, p->info) && p->info != a)
      printf("%d ", p->info);
    p = p->next;
  }
  printf("\n");
  return;
}

int main(void) {
  struct nodo *G[MAX];
  int n, a, b, c;
  n = leggiGrafo(G);
  printf("inserisci tre vertici a, b e c del grafo: ");
  scanf("%d %d %d", &a, &b, &c);
  stampaAmiciComuni(G, n, a, b, c);
  stampaAmiciNonComuni(G, n, a, c);
  return(0);
}
```

Esercizio n. 2

Dopo aver acquisito in input una matrice di ordine $n \times m$ di interi positivi, eliminare tutte le colonne la cui somma degli elementi sia dispari, spostando di una posizione verso sinistra le colonne successive ed aggiungendo una colonna di elementi nulli a destra. Stampare la matrice ottenuta al termine dell'elaborazione.

Esempio Sia $n = 3$ e $m = 5$ e sia A la matrice $n \times m$ originale (in grassetto gli elementi delle colonne di somma dispari) e A' la matrice A trasformata al termine dell'elaborazione

$$A = \begin{pmatrix} \mathbf{1} & 2 & \mathbf{3} & \mathbf{4} & 5 \\ \mathbf{17} & 12 & \mathbf{2} & \mathbf{2} & 3 \\ \mathbf{5} & 4 & \mathbf{6} & \mathbf{1} & 2 \end{pmatrix} \quad A' = \begin{pmatrix} 2 & 5 & 0 & 0 & 0 \\ 12 & 3 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 & 0 \end{pmatrix}$$

Soluzione

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

void leggiMatrice(int A[MAX][MAX], int *n, int *m) {
  int i, j;
  printf("Numero di righe e di colonne: ");
  scanf("%d %d", n, m);
  printf("Inserisci gli elementi della matrice:\n");
  for (i=0; i<*n; i++)
    for (j=0; j<*m; j++)
      scanf("%d", &A[i][j]);
  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++)
      printf("%2d ", A[i][j]);
    printf("\n");
  }
  return;
}

void eliminaColonne(int A[MAX][MAX], int n, int m) {
  int i, j, k, s;
  for (j=0; j<m; j++) {
    s = 0;
    for (i=0; i<n; i++)
      s = s + A[i][j];
    if (s % 2 == 1) {
      for (k=j+1; k<m; k++)
        for (i=0; i<n; i++)
          A[i][k-1] = A[i][k];
      for (i=0; i<n; i++)
        A[i][m-1] = 0;
      j = j-1;
    }
  }
  return;
}

int main(void) {
  int A[MAX][MAX], n, m;
  leggiMatrice(A, &n, &m);
  eliminaColonne(A, n, m);
  stampaMatrice(A, n, m);
  return(0);
}
```