Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2015/2016

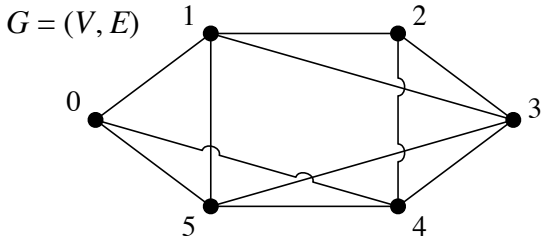
Esame scritto del 18 Gennaio 2016 (Appello A)

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 le liste di adiacenza di un grafo non orientato $G = (V, E)$, con n vertici. Stampare le liste di adiacenza di G. Letto in input un vertice $u \in V(G)$ stampare (una sola volta!) tutti i vertici $v \in V(G)$ a distanza 2 da u.

Esempio Si consideri il grafo in figura; sia $u = 0$. I vertici a distanza 2 da u sono $v = 2$ e $v = 3$.



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 n, i;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("inserisci %d interi: ", 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("%d: ", i);
49      stampaLista(G[i]);
50    }
51    return;
52  }
53
54  int adiacente(int x, struct nodo *p) {
55    int r;
56    while (p != NULL && p->info != x)
57      p = p->next;
58    if (p != NULL)
59      r = 1;
60    else
61      r = 0;
62    return(r);
63  }
64
```

```c
int main(void) {
  struct nodo *G[MAX], *p, *q, *r=NULL, *s;
  int n, u;
  n = leggiGrafo(G);
  stampaGrafo(G, n);
  printf("Inserisci un vertice u del grafo: ");
  scanf("%d", &u);
  printf("Vertici a distanza 2 da %d: ", u);
  p = G[u];
  while (p != NULL) {
    q = G[p->info];
    while (q != NULL) {
      if (q->info!=u && !adiacente(q->info,G[u]) && !adiacente(q->info,r)) {
        s = malloc(sizeof(struct nodo));
        s->info = q->info;
        s->next = r;
        r = s;
      }
      q = q->next;
    }
    p = p->next;
  }
  stampaLista(r);
  return(0);
}
```

Esercizio n. 2

Generare tre array A, B e C di numeri naturali casuali compresi nell'intervallo $[0, 10]$, rispettivamente con n, m e k elementi ciascuno. Costruire la matrice D composta da 3 righe e $n \times m \times k$ colonne, data dal prodotto cartesiano $A \times B \times C$ (ogni colonna di D è data da una combinazione di elementi di A, B e C). Stampare D e contare le colonne di D che hanno gli elementi in ordine crescente.

Esempio Sia $n = 3$, $m = 2$, $k = 2$; consideriamo i seguenti array:

$$\begin{aligned} A &= (3, 7, 1) \\ B &= (5, 1) \\ C &= (4, 6) \end{aligned}$$

La matrice D è la seguente; in grassetto sono riportate le due colonne i cui elementi sono in ordine crescente.

$$A = \begin{pmatrix} 3 & \mathbf{3} & 3 & 3 & 7 & 7 & 7 & 7 & 1 & \mathbf{1} & 1 & 1 \\ 5 & \mathbf{5} & 1 & 1 & 5 & 5 & 1 & 1 & 5 & \mathbf{5} & 1 & 1 \\ 4 & \mathbf{6} & 4 & 6 & 4 & 6 & 4 & 6 & 4 & \mathbf{6} & 4 & 6 \end{pmatrix}$$

Soluzione

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

int generaArray(int X[]) {
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  for (i=0; i<n; i++)
    X[i] = rand() % 11;
  return(n);
}

void stampaArray(int X[], int n) {
  int i;
  for (i=0; i<n; i++)
    printf("%2d ", X[i]);
  printf("\n");
  return;
}

void stampaMatrice(int X[MAX][MAX], int n, int m) {
  int i, j;
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++)
      printf("%2d ", X[i][j]);
    printf("\n");
```

```
29    }
30    printf("\n");
31    return;
32  }
33
34  int costruisciMatrice(int A[], int n, int B[], int m,
35      int C[], int k, int D[3][MAX]) {
36
37    int i, j, h, w=0, cont=0;
38    for (i=0; i<n; i++)
39      for (j=0; j<m; j++)
40        for (h=0; h<k; h++) {
41          D[0][w] = A[i];
42          D[1][w] = B[j];
43          D[2][w] = C[h];
44          if (A[i] < B[j] && B[j] < C[h])
45            cont++;
46          w++;
47        }
48    return(cont);
49  }
50
51  int main(void) {
52    int A[MAX], B[MAX], C[MAX], D[3][MAX], n, m, k, cont;
53    n = generaArray(A);
54    m = generaArray(B);
55    k = generaArray(C);
56    cont = costruisciMatrice(A, n, B, m, C, k, D);
57    printf("A: ");
58    stampaArray(A, n);
59    printf("B: ");
60    stampaArray(B, m);
61    printf("C: ");
62    stampaArray(C, k);
63    printf("D:\n");
64    stampaMatrice(D, 3, n*m*k);
65    printf("%d colonne sono crescenti.\n", cont);
66    return(0);
67  }
```