Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2014/2015

Esame scritto del 6 Febbraio 2015 (Appello B)

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

Letti in input due interi $n, m > 0$, leggere in input una una matrice A di ordine $n \times m$ (n righe ed m colonne) di numeri razionali. Stampare la matrice A. Costruire una vettore B di m elementi i cui elementi b_k siano dati dal numero di colonne di A la cui somma degli elementi sia maggiore della somma degli elementi della colonna k. Stampare il vettore B.

Esempio Sia $n = 4$ e $m = 6$; consideriamo la seguente matrice:

$$A = \begin{pmatrix} 12.4 & -5.03 & 12.0 & -6.1 & 9.67 \\ 4.0 & -2.1 & -14.007 & 59.6 & -10.14 \\ -6.3 & 36.2 & 12.8 & -37.1 & -6.54 \\ 8.2 & -11.5 & -9.367 & 3.98 & -10.01 \end{pmatrix}$$

Il vettore B di $m = 5$ elementi è quindi il seguente: $B = (1, 2, 3, 0, 4)$

Soluzione

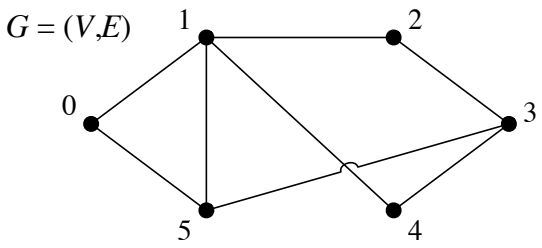
```c
#include <stdlib.h>
#include <stdio.h>

#define MAX 100

void leggiMatrice(float A[MAX][MAX], int *n, int *m) {
  int i, j;
  printf("Numero di righe e di colonne: ");
  scanf("%d %d", n, m);
  for (i=0; i<*n; i++) {
    printf("Elementi della riga %d: ", i);
    for (j=0; j<*m; j++)
      scanf("%f", &A[i][j]);
  }
  return;
}

```

```c
void stampaMatrice(float A[MAX][MAX], int n, int m) {
  int i, j;
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++)
      printf("%7.3f ", A[i][j]);
    printf("\n");
  }
  return;
}

void costruisciVettore(float A[MAX][MAX], int B[MAX], int n, int m) {
  int i, j, k;
  float s1, s2;
  for (k=0; k<m; k++) {
    B[k] = 0;
    s1 = 0;
    for (i=0; i<n; i++)
      s1 = s1+A[i][k];
    for (j=0; j<m; j++) {
      if (j != k) {
        s2 = 0;
        for (i=0; i<n; i++)
          s2 = s2 + A[i][j];
        if (s2 > s1)
          B[k]++;
      }
    }
  }
  return;
}

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

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

Esercizio n. 2

Leggere in input le liste di adiacenza di un grafo non orientato $G = (V, E)$, con n vertici. Letta in input una lista C di $k \leq n$ elementi di $V(G)$, verificare se C costituisce un insieme indipendente di vertici di G. Un insieme di vertici è *indipendente* se i vertici sono tutti non adiacenti fra di loro.

Esempio Si consideri il grafo $G = (V, E)$ rappresentato in figura. La lista $C = (0, 4, 2, 5)$ non è un insieme indipendente di vertici perché $(0, 5) \in E(G)$; al contrario $C = (2, 4, 5)$ è un insieme indipendente di vertici di G.



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);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    p->next = primo;
    primo = p;
  }
  return(primo);
}

int leggiGrafo(struct nodo *G[]) {
  int i, n;
  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  int adiacente(int x, struct nodo *p) {
46    int r;
47    while (p != NULL && p->info != x)
48      p = p->next;
49    if (p != NULL)
50      r = 1;
51    else
52      r = 0;
53    return(r);
54  }
55
56  int insiemeIndipendente(struct nodo *C, struct nodo *G[], int n) {
57    struct nodo *p, *q;
58    int flag = 1;
59    p = C;
60    while (p != NULL && flag == 1) {
61      q = C;
62      while (q != NULL && adiacente(p->info, G[q->info]) == 0)
63        q = q->next;
64      if (q != NULL)
65        flag = 0;
66      p = p->next;
67    }
68    return(flag);
69  }
70
71  int main(void) {
72    struct nodo *G[MAX], *C;
73    int n;
74    n = leggiGrafo(G);
75    C = leggiLista();
76    if (insiemeIndipendente(C, G, n) == 1)
```

```
77      printf("La lista e' un insieme indipendente.\n");
78    else
79      printf("La lista NON e' un insieme indipendente.\n");
80    return(0);
81  }
```