Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2019/2020

Esame scritto del 11 Settembre 2020 (Appello X)

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.
Deve essere consegnata solo la "bella copia" del compito scritto; su ciascun foglio deve essere riportato il ***nome****, il* ***cognome*** *e il* ***numero di matricola*** *(o un altro codice identificativo di fantasia) dello studente.*

Esercizio n. 1

Letti in input due interi positivi n e m generare e stampare una matrice A di $n \times m$ numeri razionali casuali compresi tra 0 e 1 (si ricorda che la funzione `rand()` produce numeri casuali interi compresi tra 0 e `RAND_MAX`, costante definita nella libreria standard). Stampare gli elementi della colonna contenente il massimo numero di elementi consecutivi decrescenti.

Esempio Siano $n = 7$ e $m = 4$ e si consideri la seguente matrice casuale

$$A = \begin{pmatrix} 0,523 & 0,7 & 0,123 & 0,0009 \\ 0,47 & \mathbf{1,0} & 0,88 & 0,42 \\ 0,9012 & \mathbf{0,63} & 0,0 & 0,89 \\ 0,17 & \mathbf{0,601} & 0,54 & 0,2 \\ 0,26 & \mathbf{0,57} & 0,41 & 0,73 \\ 0,3 & \mathbf{0,407} & 0,92 & 0,03 \\ 0,68 & 0,879 & 0,1 & 0,28 \end{pmatrix}$$

La colonna con il massimo numero di elementi decrescenti consecutivi è la seconda.

Soluzione

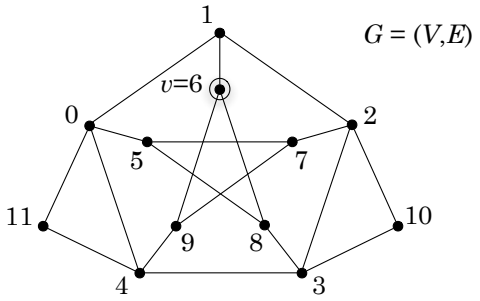
```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 50

void stampaColonna(float A[MAX][MAX], int n, int j) {
  int i;
  printf("Elementi della colonna %d:\n", j);
  for (i=0; i<n; i++)
    printf("%1.5f\n", A[i][j]);
  return;
}

void generaMatrice(float 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] = (float)rand()/RAND_MAX;
```

```
  return;
}

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("%1.6f ", A[i][j]);
    printf("\n");
  }
  return;
}

int trovaColonna(float A[MAX][MAX], int n, int m) {
  int i, j, cont, contBis, contMax, jMax;
  contMax = 0;
  jMax = -1;
  for (j=0; j<m; j++) {
    cont = 0;
    contBis = 0;
    for (i=0; i<n-1; i++) {
      if (A[i][j] > A[i+1][j]) {
        cont++;
      } else {
        if (cont > contBis)
          contBis = cont;
        cont = 0;
      }
    }
    if (cont > contBis)
      contBis = cont;
    if (contBis > contMax) {
      contMax = contBis;
      jMax = j;
    }
  }
  return(jMax);
}

int main(void) {
  float A[MAX][MAX];
  int n, m, j;
  printf("Numero di righe e colonne: ");
  scanf("%d %d", &n, &m);
  generaMatrice(A, n, m);
  stampaMatrice(A, n, m);
  j = trovaColonna(A, n, m);
  if (j > -1)
    stampaColonna(A, n, j);
  else
    printf("Nessuna colonna presenta elementi consecutivi decrescenti.\n");
  return(1);
}
```

Esercizio n. 2

Letto un grafo non orientato $G = (V, E)$ con n vertici, rappresentarlo con liste di adiacenza. Letto inoltre in input un vertice $v \in V(G)$ costruire la lista L dei vertici $u \in V(G)$ a distanza 2 da v; in altri termini si chiede di costruire la lista di tutti i vertici u non adiacenti a v, ma adiacenti ad un vertice adiacente a v. Stampare la lista L.

Esempio Si consideri il grafo $G = (V, E)$ rappresentato in figura.



Sia $v = 6$ il vertice letto in input. La lista L dei vertici a distanza 2 da v è la seguente:

$$L : 0 \to 2 \to 3 \to 4 \to 5 \to 7 \to \text{NULL}$$

Soluzione

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

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

struct nodo *leggiLista(void) {
  struct nodo *p, *primo=NULL;
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d elementi: ", 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 del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
```

```
30      printf("Lista dei vertici adiacenti al vertice %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 esiste(struct nodo *p, int x) {
55    int rc;
56    while (p != NULL && p->info != x)
57      p = p->next;
58    if (p == NULL)
59      rc = 0;
60    else
61      rc = 1;
62    return(rc);
63  }
64
65  int adiacenti(struct nodo *G[], int v, int u) {
66    int rc;
67    if (esiste(G[v], u))
68      rc = 1;
69    else
70      rc = 0;
71    return(rc);
72  }
73
74  struct nodo *costruisciLista(struct nodo *G[], int v) {
75    struct nodo *p, *q, *r, *primo=NULL;
76    p = G[v];
77    while (p != NULL) {
78      q = G[p->info];
79      while (q != NULL) {
80        if (q->info != v && !adiacenti(G, v, q->info) &&
81          !esiste(primo, q->info)) {
82          r = malloc(sizeof(struct nodo));
83          r->info = q->info;
84          r->next = primo;
85          primo = r;
```

```
86        }
87        q = q->next;
88      }
89      p = p->next;
90    }
91    return(primo);
92  }
93
94  int main(void) {
95    struct nodo *G[MAX], *L;
96    int v, n;
97    n = leggiGrafo(G);
98    printf("Inserisci un vertice del grafo: ");
99    scanf("%d", &v);
100   L = costruisciLista(G, v);
101   printf("Liste di adiacenza del grafo G:\n");
102   stampaGrafo(G, n);
103   printf("Lista dei vertici a distanza 2 da %d:\n", v);
104   stampaLista(L);
105   return(0);
106 }
```