Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2021/2022

Esame scritto del 11 Febbraio 2022 (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.
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

Letto in input il numero intero $n > 0$, generare una matrice quadrata A di ordine n composta da numeri interi casuali in $\{0, 1\}$. Stampare la matrice A. Verificare se la matrice è simmetrica rispetto alla diagonale principale, alla diagonale secondaria, rispetto all'asse verticale e orizzontale.

Esempio Sia $n = 5$ e si considerino le seguenti matrici di numeri casuali:

$$\mathscr{A} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \quad \mathscr{B} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} \quad \mathscr{C} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

La matrice $\mathscr{A}$ è simmetrica rispetto alla diagonale secondaria, la matrice $\mathscr{B}$ è simmetrica rispetto all'asse verticale e la matrice $\mathscr{C}$ è simmetrica rispetto alla diagonale principale.

Soluzione

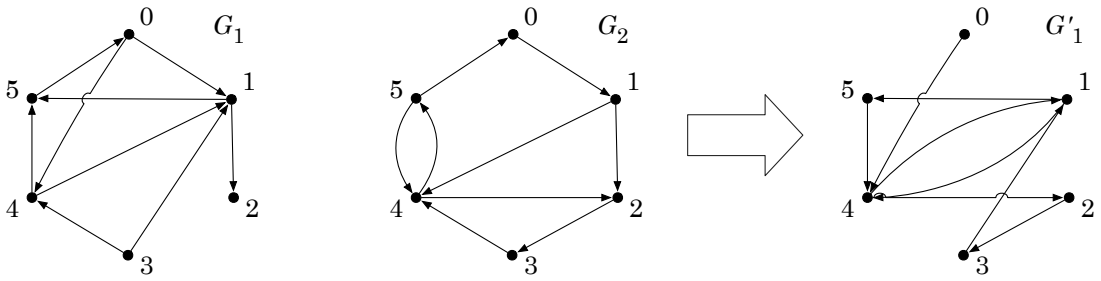
```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 50

void matriceBinaria(int A[MAX][MAX], int n) {
  for (int i=0; i<n; i++)
    for (int j=0; j<n; j++)
      A[i][j] = rand() % 2;
  return;
}

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

```c

int principale(int A[MAX][MAX], int n) {
  int i, j, flag = 1;
  for (i=0; i < n && flag == 1; i++)
    for (j=i; j < n && flag == 1; j++)
      if (A[i][j] != A[j][i])
        flag = 0;
  return(flag);
}

int secondaria(int A[MAX][MAX], int n) {
  int i, j, flag = 1;
  for (i=0; i < n && flag == 1; i++)
    for (j=0; j < n && flag == 1; j++)
      if (A[i][j] != A[n-j-1][n-i-1])
        flag = 0;
  return(flag);
}

int verticale(int A[MAX][MAX], int n) {
  int i, j, flag = 1;
  for (i=0; i < n && flag == 1; i++)
    for (j=0; j < n/2 && flag == 1; j++)
      if (A[i][j] != A[i][n-j-1])
        flag = 0;
  return(flag);
}

int orizzontale(int A[MAX][MAX], int n) {
  int i, j, flag = 1;
  for (i=0; i < n/2 && flag == 1; i++)
    for (j=0; j < n && flag == 1; j++)
      if (A[i][j] != A[n-i-1][j])
        flag = 0;
  return(flag);
}

int main(void) {
  int A[MAX][MAX], n;
  srand((unsigned)time(NULL));
  printf("Ordine della matrice: ");
  scanf("%d", &n);
  matriceBinaria(A, n);
  stampaMatrice(A, n);
  if (principale(A,n) == 1)
    printf("La matrice e' simmetrica rispetto alla diagonale principale\n");
  if (secondaria(A,n) == 1)
    printf("La matrice e' simmetrica rispetto alla diagonale secondaria\n");
  if (verticale(A,n) == 1)
    printf("La matrice e' simmetrica rispetto all'asse verticale\n");
  if (orizzontale(A,n) == 1)
    printf("La matrice e' simmetrica rispetto all'asse orizzontale\n");
  return(0);
}
```

Esercizio n. 2

Letto in input un grafo orientato $G = (V, E)$ con n vertici, rappresentarlo con liste di adiacenza. Indichiamo con e_v il grado entrante di ciascun vertice $v \in V(G)$. Visualizzare in output una coppia di vertici $u, v \in V(G)$ tale che $u \neq v$ e $|e_u - e_v|$ sia massimo.

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



La coppia di vertici con la massima differenza di grado entrante è data da $u = 1$ con $e_1 = 3$ e $v = 4$ con $e_4 = 0$.

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++) {
    printf("Lista dei vertici adiacenti al vertice %d:\n", i);
    G[i] = leggiLista();
  }
```

```
33    return(n);
34  }
35
36  int adiacente(struct nodo *G[], int u, int v) {
37    struct nodo *p;
38    int flag;
39    p = G[u];
40    while (p != NULL && p->info != v)
41      p = p->next;
42    if (p == NULL)
43      flag = 0;
44    else
45      flag = 1;
46    return(flag);
47  }
48
49  void addSpigolo(struct nodo *G[], int u, int v) {
50    struct nodo *p;
51    p = malloc(sizeof(struct nodo));
52    p->info = v;
53    p->next = G[u];
54    G[u] = p;
55    return;
56  }
57
58  void delSpigolo(struct nodo *G[], int u, int v) {
59    struct nodo *p, *q=NULL;
60    p = G[u];
61    while (p != NULL && p->info != v) {
62      q = p;
63      p = p->next;
64    }
65    if (p != NULL) {
66      if (q == NULL)
67        G[u] = G[u]->next;
68      else
69        q->next = p->next;
70      free(p);
71    }
72    return;
73  }
74
75  void stampaLista(struct nodo *p) {
76    while (p != NULL) {
77      printf("%d --> ", p->info);
78      p = p->next;
79    }
80    printf("NULL\n");
81    return;
82  }
83
84  void stampaGrafo(struct nodo *G[], int n) {
85    int i;
86    for (i=0; i<n; i++) {
87      printf("%d: ", i);
88      stampaLista(G[i]);
```

```
  }
  return;
}

int main(void) {
  struct nodo *G1[MAX], *G2[MAX], *p;
  int n, i;
  n = leggiGrafo(G1);
  n = leggiGrafo(G2);
  for (i=0; i<n; i++) {
    p = G2[i];
    while (p != NULL) {
      if (adiacente(G1, i, p->info))
        delSpigolo(G1, i, p->info);
      else
        addSpigolo(G1, i, p->info);
      p = p->next;
    }
  }
  stampaGrafo(G1, n);
  return(0);
}
```