Corso di Algoritmi e Strutture Dati (IN110) – Prof. Marco Liverani – a.a. 2023/2024

Seconda prova di esonero – 10 gennaio 2024

Risolvere i seguenti problemi proponendo, per ciascun esercizio, la codifica in linguaggio C di un programma completo. La prova dura tre ore, durante le quali non è possibile allontanarsi dall'aula, se non dopo aver consegnato l'elaborato scritto. Per superare la prova di esonero è necessario ottenere almeno 15 punti; tuttavia affinché le prove di esonero siano valide è necessario che la media dei voti del primo e del secondo esonero sia maggiore o uguale a 18/30. È possibile utilizzare libri e appunti personali, senza scambiarli con altri studenti. I compiti che presenteranno evidenti ed anomale "similitudini" saranno annullati.

Esercizio n. 1

Letti in input due numeri interi $n, m > 0$ tali che $n > m$, generare due liste L_1 e L_2 di interi casuali in $\{0, 1\}$, rispettivamente di n e di m elementi ciascuna. Stampare le due liste. Contare il numero di occorrenze della lista L_2 nella lista L_1.

Esempio Siano $n = 11$ e $m = 3$ i due interi letti in input. Supponiamo che siano state generate le seguenti liste di numeri casuali in $\{0, 1\}$:

$$\begin{aligned} L_1 &= 0 \to 1 \to 1 \to \mathbf{0} \to \mathbf{1} \to \mathbf{0} \to 0 \to \mathbf{0} \to \mathbf{1} \to \mathbf{0} \to 1 \\ L_2 &= 0 \to 1 \to 0 \end{aligned}$$

La lista L_2 compare due volte in L_1.

Soluzione

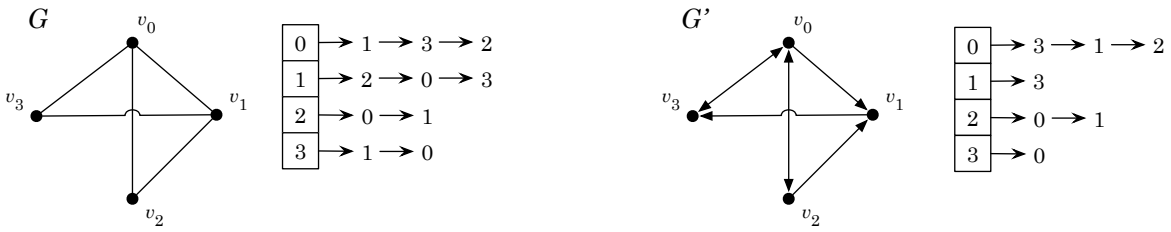
```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

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

void stampaLista(struct nodo *p) {
  while (p != NULL) {
    printf("%d --> ", p->info);
    p = p->next;
  }
  printf("NULL\n");
  return;
```

```
}

int conta(struct nodo *L1, struct nodo *L2) {
  struct nodo *p, *p1, *q;
  int cont = 0;
  p = L1;
  while (p != NULL) {
    q = L2;
    p1 = p;
    while (q != NULL && p1 != NULL && q->info == p1->info) {
      p1 = p1->next;
      q = q->next;
    }
    if (q == NULL)
      cont = cont+1;
    p = p->next;
  }
  return(cont);
}

struct nodo *costruisciLista(void) {
  struct nodo *p, *primo = NULL;
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->next = primo;
    p->info = rand() % 2;
    primo = p;
  }
  return(primo);
}

int main(void) {
  struct nodo *L1, *L2;
  int c;
  srand((unsigned)time(NULL));
  L1 = costruisciLista();
  L2 = costruisciLista();
  stampaLista(L1);
  stampaLista(L2);
  c = conta(L1, L2);
  printf("L2 e' contenuta in L1 %d volte.\n", c);
  return(0);
}
```

Esercizio n. 2

Un grafo $G = (V, E)$ è *non orientato* se per ogni $(u, v) \in E$ risulta anche $(v, u) \in E$. Letto in input un grafo G rappresentarlo con liste di adiacenza. Stampare in output le liste di adiacenza di G. Verificare se G è orientato o non orientato.

Esempio In figura sono rappresentati il grafo G non orientato e il grafo G' orientato.



Soluzione

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

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

void stampaLista(struct nodo *p) {
  while (p != NULL) {
    printf("%d --> ", p->info);
    p = p->next;
  }
  printf("NULL\n");
  return;
}

void stampaGrafo(struct nodo *G[], int n) {
  int i;
  for (i=0; i<n; i++) {
    printf("%2d: ", i);
    stampaLista(G[i]);
  }
  return;
}

```

```
28 int adiacenti(struct nodo *G[], int u, int v) {
29   struct nodo *p;
30   int r;
31   p = G[u];
32   while (p != NULL && p->info != v)
33     p = p->next;
34   if (p == NULL)
35     r = 0;
36   else
37     r = 1;
38   return(r);
39 }
40
41 int orientato(struct nodo *G[], int n) {
42   int flag = 0, u;
43   struct nodo *p;
44   for (u=0; u < n && flag == 1; u++) {
45     p = G[u];
46     while (p != NULL) {
47       if (!adiacenti(G, p->info, u))
48         flag = 1;
49       p = p->next;
50     }
51   }
52   return(flag);
53 }
54
55 struct nodo *leggiLista(void) {
56   struct nodo *p, *primo = NULL;
57   int i, n;
58   printf("Numero di elementi: ");
59   scanf("%d", &n);
60   printf("Elementi della lista: ");
61   for (i=0; i<n; i++) {
62     p = malloc(sizeof(struct nodo));
63     scanf("%d", &p->info);
64     p->next = primo;
65     primo = p;
66   }
67   return(primo);
68 }
69
70 int leggiGrafo(struct nodo *G[]) {
71   int i, n;
72   printf("Numero di vertici del grafo: ");
73   scanf("%d", &n);
74   for (i=0; i<n; i++) {
```

```
75      printf("Lista di adiacenza del vertice %d\n", i);
76      G[i] = leggiLista();
77    }
78    return(n);
79  }
80
81  int main(void) {
82    struct nodo *G[MAX];
83    int n;
84    n = leggiGrafo(G);
85    stampaGrafo(G, n);
86    if (orientato(G, n))
87      printf("Il grafo e' orientato\n");
88    else
89      printf("Il grafo NON e' orientato\n");
90    return(0);
91  }
```