Corso di Algoritmi e Strutture Dati (IN110) – Prof. Marco Liverani – a.a. 2022/2023

Seconda prova di esonero – 13 gennaio 2023

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

Letto in input un intero $n > 0$ costruire una lista con n elementi costituiti da due campi numerici e un puntatore all'elemento successivo: il primo è un numero casuale compreso tra 0 e 10 (estremi inclusi), il secondo è il numero di volte che tale valore casuale si ripete nella lista. Stampare in output la lista.

Esempio Sia $n = 8$; una possibile lista prodotta dal programma può essere la seguente:

$$(3,1) \to (5,2) \to (7,3) \to (7,3) \to (5,2) \to (10,1) \to (7,3) \to (6,1)$$

Soluzione

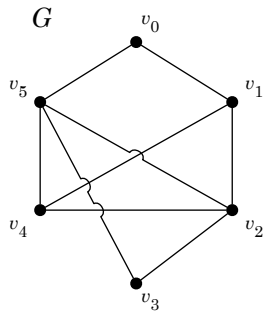
```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 100

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

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

```

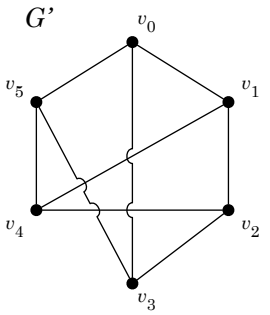
```
20 struct nodo *costruisciLista(void) {
21   struct nodo *p, *q, *primo = NULL;
22   int i, n;
23   printf("Numero di elementi: ");
24   scanf("%d", &n);
25   for (i=0; i<n; i++) {
26     p = malloc(sizeof(struct nodo));
27     p->next = primo;
28     p->info = rand() % 11;
29     primo = p;
30   }
31   while (p != NULL) {
32     p->ripetizioni = 0;
33     q = primo;
34     while (q != NULL) {
35       if (q->info == p->info)
36         p->ripetizioni++;
37       q = q->next;
38     }
39     p = p->next;
40   }
41   return(primo);
42 }
43
44 int main(void) {
45   struct nodo *p;
46   p = costruisciLista();
47   stampaLista(p);
48   return(0);
49 }
```

Esercizio n. 2

Lette in input le liste di adiacenza di un grafo non orientato $G = (V, E)$, verificare se G è k-regolare. In caso affermativo visualizzare il valore di k. Un grafo si dice k-regolare, se tutti i suoi vertici hanno grado k.

Esempio Si considerino il grafo G e il grafo G' rappresentati in figura. Il grafo G non è regolare, perché ad esempio il v_0 ha grado 2, mentre il vertice v_1 ha grado 3; viceversa il grafo G' è k-regolare con $k = 3$, perché tutti i suoi vertici hanno grado 3.





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]);
```

```
24    }
25    return;
26  }
27
28  struct nodo *leggiLista() {
29    struct nodo *p, *primo = NULL;
30    int i, n;
31    printf("Numero di elementi: ");
32    scanf("%d", &n);
33    printf("Elementi della lista: ");
34    for (i=0; i<n; i++) {
35      p = malloc(sizeof(struct nodo));
36      scanf("%d", &p->info);
37      p->next = primo;
38      primo = p;
39    }
40    return(primo);
41  }
42
43  int leggiGrafo(struct nodo *G[]) {
44    int i, n;
45    printf("Numero di vertici del grafo: ");
46    scanf("%d", &n);
47    for (i=0; i<n; i++) {
48      printf("Lista di adiacenza del vertice %d\n", i);
49      G[i] = leggiLista();
50    }
51    return(n);
52  }
53
54  int grado(struct nodo *p) {
55    int g = 0;
56    while (p != NULL) {
57      g++;
58      p = p->next;
59    }
60    return(g);
61  }
62
63  int grafoRegolare(struct nodo *G[], int n) {
64    int i, k;
65    k = grado(G[0]);
66    for (i=1; i<n && grado(G[i]) == k; i++) ;
67    if (i<n)
68      k = -1;
69    return(k);
70  }
```

```
71
72 int main(void) {
73   struct nodo *G[MAX];
74   int n, k;
75   n = leggiGrafo(G);
76   stampaGrafo(G, n);
77   k = grafoRegolare(G, n);
78   if (k >= 0)
79     printf("Il grafo e' %d-regolare\n", k);
80   else
81     printf("Il grafo non e' k-regolare\n");
82   return(0);
83 }
```