Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2009/2010

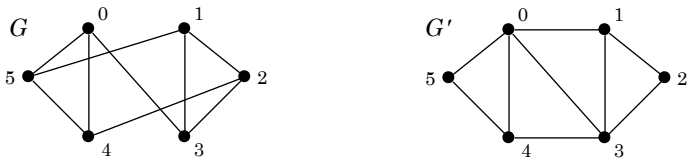
Seconda prova di esonero – 8 gennaio 2010

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 grafo non orientato $G = (V, E)$, rappresentarlo con liste di adiacenza. Letto in input un numero intero $k > 0$ verificare se G è un grafo k-regolare. Un grafo si dice k-regolare se il grado di tutti i suoi vertici è k.

Esempio Si considerino i grafi rappresentati in figura. Il grafo G è k-regolare, con $k = 3$, mentre il grafo G' non è k-regolare, visto che ha vertici di grado 2, 3 e 4.



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 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);
}
```

```
24
25 int leggiGrafo(struct nodo *G[]) {
26   int n, i;
27   printf("Numero di vertici del grafo: ");
28   scanf("%d", &n);
29   for (i=0; i<n; i++) {
30     printf("\nLista di adiacenza del vertice %d:\n", i);
31     G[i] = leggiLista();
32   }
33   return(n);
34 }
35
36 int regolare(struct nodo *G[], int n, int k) {
37   int i, c, flag = 1;
38   struct nodo *p;
39   for (i=0; i<n && flag; i++) {
40     c = 0;
41     p = G[i];
42     while (p!=NULL) {
43       c++;
44       p = p->next;
45     }
46     if (c != k)
47       flag = 0;
48   }
49   return(flag);
50 }
51
52 int main(void) {
53   struct nodo *G[MAX];
54   int n, k;
55   n = leggiGrafo(G);
56   printf("Inserisci un intero positivo: ");
57   scanf("%d", &k);
58   if (regolare(G, n, k))
59     printf("Il grafo e' %d-regolare.\n", k);
60   else
61     printf("Il grafo non e' %d-regolare.\n", k);
62   return(0);
63 }
```

Esercizio n. 2

Letti in input due interi positivi n e k, costruire una lista A di n numeri casuali interi positivi minori di k. Stampare la lista A. Eliminare dalla lista A tutti gli elementi duplicati, memorizzandoli in una nuova lista B. Stampare le liste A e B.

Esempio Supponiamo che $n = 10$ e $k = 5$ e che la lista A di numeri casuali minori di k sia la seguente:

$$A : 2 \to 4 \to 0 \to 4 \to 3 \to 3 \to 1 \to 3 \to 2 \to 1 \to \text{ null}$$

Al termine dell'elaborazione il programma produce le seguenti liste:

$$A : 2 \to 4 \to 0 \to 3 \to 1 \to \text{ null}$$

$$B : 1 \to 3 \to 3 \to 4 \to 2 \to \text{ null}$$

Si noti che è ininfluente l'ordine degli elementi delle due liste: possono essere costruite in un ordine arbitrario.

Soluzione

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

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

struct nodo *doppioni(struct nodo *p) {
  struct nodo *q, *r=NULL, *s, *t;
  while (p != NULL) {
    s = p;
    while (s->next != NULL) {
      if (s->next->info == p->info) {
        t = s->next;
        q = malloc(sizeof(struct nodo));
        q->info = p->info;
        q->next = r;
        r = q;
        s->next = t->next;
        free(t);
      } else {
        s = s->next;
      }
    }
    p = p->next;
  }
  return(r);
}

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

```
40
41 struct nodo *listaRandom(void) {
42   struct nodo *p, *primo = NULL;
43   int i, k, n;
44   srand((unsigned)time(NULL));
45   printf("Numero di elementi: ");
46   scanf("%d", &n);
47   printf("Soglia dei numeri casuali: ");
48   scanf("%d", &k);
49   for (i=0; i<n; i++) {
50     p = malloc(sizeof(struct nodo));
51     p->info = rand() % k;
52     p->next = primo;
53     primo = p;
54   }
55   return(primo);
56 }
57
58 int main(void) {
59   struct nodo *p1, *p2;
60   p1 = listaRandom();
61   printf("Lista originale: ");
62   stampaLista(p1);
63   p2 = doppioni(p1);
64   printf("Lista ridotta: ");
65   stampaLista(p1);
66   printf("Lista dei doppioni: ");
67   stampaLista(p2);
68   return(0);
69 }
```