Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2015/2016

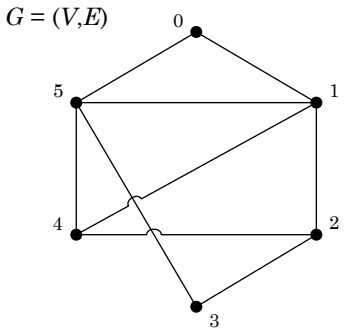
Seconda prova di esonero – 8 gennaio 2016

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.
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

Lette in input le liste di adiacenza di un grafo non orientato $G = (V, E)$ con n vertici, selezionare in modo casuale un vertice $v \in V(G)$ e, se v non è un vertice isolato, stampare il rapporto $k/|N(v)|$, dove $|N(v)|$ è il numero di vertici adiacenti a v e $k = |\{(u, w) : (u, w) \in E(G) \text{ e } u, w \in N(v)\}|$.

Esempio Sia $n = 6$ e si consideri il grafo G in figura. Per $v = 5$ risulta $|N(v)| = 4$ e $k = 2$ (risulta, infatti, che $(0, 1), (1, 4) \in E$); quindi $k/|N(v)| = 0,5$.



Soluzione

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

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

void stampaLista(struct nodo *p) {
  while (p!=NULL) {
```

```
13      printf("%d --> ", p->info);
14      p = p->next;
15    }
16    printf("NULL\n");
17    return;
18  }
19
20  void stampaGrafo(struct nodo *G[], int n) {
21    printf("\nListe di adiacenza del grafo:\n");
22    for (int i=0; i<n; i++) {
23      printf(" %2d: ", i);
24      stampaLista(G[i]);
25    }
26    return;
27  }
28
29  struct nodo *leggiLista(void) {
30    struct nodo *p, *primo=NULL;
31    int n, i;
32    printf(" Numero di elementi della lista: ");
33    scanf("%d", &n);
34    printf(" Elementi della lista: ");
35    for (i=0; i<n; i++) {
36      p = malloc(sizeof(struct nodo));
37      scanf("%d", &p->info);
38      p->next = primo;
39      primo = p;
40    }
41    return(primo);
42  }
43
44  int leggiGrafo(struct nodo *V[]) {
45    int i, n;
46    printf("Numero di vertici del grafo: ");
47    scanf("%d", &n);
48    for (i=0; i<n; i++) {
49      printf("\nLista di adiacenza del vertice %d:\n", i);
50      V[i] = leggiLista();
51    }
52    return(n);
53  }
54
55  int adiacente(int u, int v, struct nodo *V[]) {
56    struct nodo *p;
57    int rc;
58    p = V[u];
59    while (p != NULL && p->info != v)
60      p = p->next;
61    if (p == NULL)
62      rc = 0;
63    else
64      rc = 1;
65    return(rc);
66  }
```

```
67
68  int calcolaN(struct nodo *p) {
69    int n=0;
70    while (p != NULL) {
71      n++;
72      p = p->next;
73    }
74    return(n);
75  }
76
77  int calcolaK(struct nodo *G[], int v) {
78    struct nodo *p, *q;
79    int k=0;
80    p = G[v];
81    while (p != NULL) {
82      q = p->next;
83      while (q != NULL) {
84        if (adiacente(p->info, q->info, G) == 1)
85          k++;
86        q = q->next;
87      }
88      p = p->next;
89    }
90    return(k);
91  }
92
93  int main(void) {
94    int n, v, N, k;
95    struct nodo *G[MAX];
96    n = leggiGrafo(G);
97    stampaGrafo(G, n);
98    srand((unsigned)time(NULL));
99    v = rand() % n;
100   N = calcolaN(G[v]);
101   k = calcolaK(G, v);
102   if (N==0)
103     printf("Il vertice %d e' isolato.\n", v);
104   else
105     printf("Per v=%d risulta |N(%d)| = %d e k = %d ==> k/|N(%d)| = %.1f\n",
106       v, v, N, k, v, (float)k/N);
107   return(0);
108 }
```

Esercizio n. 2

Lette in input le liste di adiacenza di un grafo non orientato $G = (V, E)$ con n vertici, costruire il vettore C, assegnando, per $i = 0, 1, \ldots, n-1$, a C_i il numero di vertici di G con grado i.

Esempio Facendo riferimento al grafo G rappresentato nella figura precedente, risulta

$$C_0 = 0, C_1 = 0, C_2 = 2, C_3 = 2, C_4 = 2, C_5 = 0$$

Soluzione

```
#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;
  printf("\nListe di adiacenza del grafo:\n");
  for (i=0; i<n; i++) {
    printf(" %2d: ", i);
    stampaLista(G[i]);
  }
  return;
}

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

```

```
44 int leggiGrafo(struct nodo *V[]) {
45   int i, n;
46   printf("Numero di vertici del grafo: ");
47   scanf("%d", &n);
48   for (i=0; i<n; i++) {
49     printf("\nLista di adiacenza del vertice %d:\n", i);
50     V[i] = leggiLista();
51   }
52   return(n);
53 }
54 
55 int calcolaGrado(struct nodo *p) {
56   int n=0;
57   while (p != NULL) {
58     n++;
59     p = p->next;
60   }
61   return(n);
62 }
63 
64 void vettoreGrado(int C[], struct nodo *G[], int n) {
65   int i;
66   for (i=0; i<n; i++)
67     C[i] = 0;
68   for (i=0; i<n; i++)
69     C[calcolaGrado(G[i])]++;
70   return;
71 }
72 
73 int main(void) {
74   int i, n, C[MAX];
75   struct nodo *G[MAX];
76   n = leggiGrafo(G);
77   stampaGrafo(G, n);
78   vettoreGrado(C, G, n);
79   for (i=0; i<n; i++)
80     printf("%2d vertici di grado %d\n", C[i], i);
81   return(0);
82 }
```