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

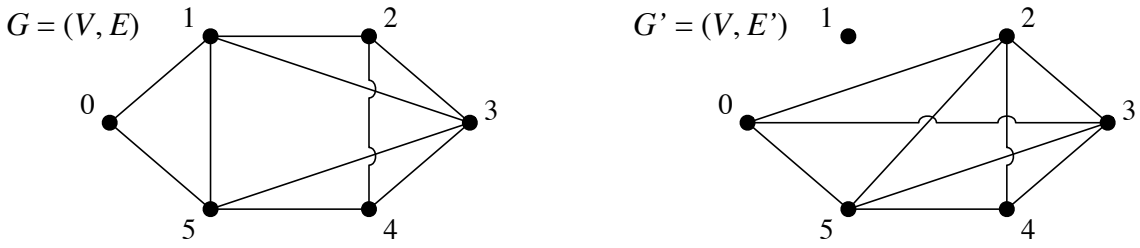
Esame scritto del 10 Febbraio 2016 (Appello B)

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. Si richiede di riportare sul foglio del compito il proprio nominativo completo ed il numero di matricola o un codice identificativo personale equivalente.

Esercizio n. 1

Leggere in input le liste di adiacenza di un grafo non orientato $G = (V, E)$, con n vertici. Stampare le liste di adiacenza di G. Scelto un vertice $v \in V(G)$ di grado massimo, eliminare tutti gli spigoli incidenti su v, ed unire con uno spigolo tutti i vertici adiacenti a v (se non sono già adiacenti fra di loro). Stampare le liste di adiacenza del grafo così ottenuto.

Esempio Si consideri il grafo rappresentato sulla sinistra in figura; sia $v = 1$ il vertice scelto fra quelli di grado massimo. Il grafo ottenuto è rappresentato sulla destra.



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 n, i;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("inserisci %d interi: ", n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
```

```
19      p->next = primo;
20      primo = p;
21    }
22    return(primo);
23  }
24
25  int leggiGrafo(struct nodo *G[]) {
26    int i, n;
27    printf("Numero di vertici: ");
28    scanf("%d", &n);
29    for (i=0; i<n; i++) {
30      printf("Lista di adiacenza di %d.\n", i);
31      G[i] = leggiLista();
32    }
33    return(n);
34  }
35
36  void stampaLista(struct nodo *p) {
37    while (p != NULL) {
38      printf("%d --> ", p->info);
39      p = p->next;
40    }
41    printf("NULL\n");
42    return;
43  }
44
45  void stampaGrafo(struct nodo *G[], int n) {
46    for (int i=0; i<n; i++) {
47      printf("%d: ", i);
48      stampaLista(G[i]);
49    }
50    return;
51  }
52
53  int adiacente(struct nodo *G[], int x, int y) {
54    int r;
55    struct nodo *p;
56    p = G[x];
57    while (p != NULL && p->info != y)
58      p = p->next;
59    if (p != NULL)
60      r = 1;
61    else
62      r = 0;
63    return(r);
64  }
65
66  int grado(struct nodo *p) {
67    int g = 0;
68    while (p != NULL) {
69      g++;
70      p = p->next;
71    }
72    return(g);
73  }
74
```

```
int gradoMax(struct nodo *G[], int n) {
  int i, max = 0, gMax;
  gMax = grado(G[0]);
  for (i=1; i<n; i++)
    if (grado(G[i]) > gMax) {
      max = i;
      gMax = grado(G[i]);
    }
  return(max);
}

void addEdge(struct nodo *G[], int u, int v) {
  struct nodo *p;
  p = malloc(sizeof(struct nodo));
  p->info = v;
  p->next = G[u];
  G[u] = p;
  return;
}

void removeEdge(struct nodo *G[], int u, int v) {
  struct nodo *p, *prec = NULL;
  p = G[u];
  while (p != NULL && p->info != v) {
    prec = p;
    p = p->next;
  }
  if (prec == NULL) {
    G[u] = G[u]->next;
  } else {
    prec->next = p->next;
    free(p);
  }
  return;
}

int main(void) {
  struct nodo *G[MAX], *p;
  int n, v;
  n = leggiGrafo(G);
  stampaGrafo(G, n);
  v = gradoMax(G, n);
  printf("\n\nv = %d\n\n", v);
  while (G[v] != NULL) {
    p = G[v]->next;
    while (p != NULL) {
      if (!adiacente(G, G[v]->info, p->info)) {
        addEdge(G, G[v]->info, p->info);
        addEdge(G, p->info, G[v]->info);
      }
      p = p->next;
    }
    removeEdge(G, G[v]->info, v);
    G[v] = G[v]->next;
  }
  stampaGrafo(G, n);
```

```
  return(0);
}
```

Esercizio n. 2

Letta in input una matrice A di numeri interi con n righe ed m colonne e un array$B$ di $k < m$ interi, stampare tutte le righe di A che contengono B come sotto-array.

Esempio Siano $n = 5$, $m = 7$, $k = 3$ e consideriamo la seguente matrice A e il vettore $B = (3, 8, 5)$; le righe di A che contengono B come sotto-array sono la seconda e la quarta:

$$A = \begin{pmatrix} 10 & 4 & 7 & 3 & 8 & 4 & 5 \\ 2 & 1 & \mathbf{3} & \mathbf{8} & \mathbf{5} & 5 & 9 \\ 1 & 17 & 6 & 5 & 4 & 11 & 6 \\ 3 & 7 & 3 & 8 & \mathbf{3} & \mathbf{8} & \mathbf{5} \\ 1 & 3 & 8 & 2 & 4 & 12 & 16 \end{pmatrix}$$

Soluzione

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

int leggiArray(int A[]) {
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d numeri interi: ", n);
  for (i=0; i<n; i++)
    scanf("%d", &A[i]);
  return(n);
}

void stampaArray(int A[], int n) {
  int i;
  for (i=0; i<n; i++)
    printf("%2d ", A[i]);
  printf("\n");
  return;
}

void leggiMatrice(int A[MAX][MAX], int *n, int *m) {
  int i, j;
  printf("Numero di righe e di colonne: ");
  scanf("%d %d", n, m);
  for (i = 0; i<*n; i++) {
    printf("Riga n. %d: ", i);
    for (j=0; j<*m; j++)
      scanf("%d", &A[i][j]);
  }
  return;
}

```

```
36 int contiene(int A[], int B[], int m, int k) {
37   int i, j=0, rc=0;
38   for (i=0; i<=m-k && j<k; i++) {
39     j = 0;
40     while (j<k && A[i+j] == B[j])
41       j++;
42   }
43   if (j == k)
44     rc = 1;
45   return(rc);
46 }
47
48 int main(void) {
49   int A[MAX][MAX], B[MAX], n, m, i, k;
50   leggiMatrice(A, &n, &m);
51   k = leggiArray(B);
52   for (i=0; i<n; i++) {
53     if (contiene(A[i], B, m, k))
54       stampaArray(A[i], m);
55   }
56   return(0);
57 }
```