Corso di Informatica 1 (IN1) – Prof. Marco Liverani – a.a. 2005/2006

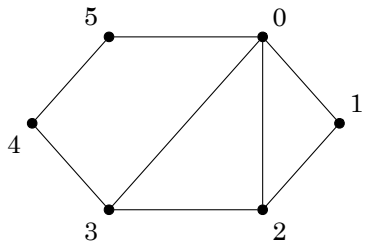
Seconda prova di esonero – 9 gennaio 2006

Esercizio n. 1

Risolvere il seguente problema proponendo una codifica completa del programma in linguaggio C.

Sia $G = (V, E)$ un grafo non orientato letto in input e rappresentato con liste di adiacenza. Letta in input una sequenza di vertici $L = (v_1, v_2, \ldots, v_k)$ di $V(G)$ (rappresentata con una lista), con $v_i \neq v_j \ \forall i \neq j$, verificare se L costituisce un cammino su G privo di corde, ossia un cammino semplice tale che $(v_i, v_j) \notin E(G) \ \forall j > i + 1$.

Esempio Si consideri il grafo $G = (V, E)$ rappresentato in figura, con $V = \{0, 1, 2, 3, 4, 5\}$ ed $E = \{(0, 1), (0, 2), (0, 3), (0, 5), (1, 2), (2, 3), (3, 4), (4, 5)\}$. La lista di vertici $(5, 0, 3, 2, 1)$ non costituisce un cammino privo di corde, infatti, pur essendo un cammino, nel grafo esistono gli spigoli $(0, 2)$ e $(0, 1)$ che rappresentano delle corde per il cammino. La lista di vertici $(5, 1, 2, 4)$ non rappresenta un cammino, perché gli spigoli $(5, 1)$ e $(2, 4)$ non sono presenti nel grafo. Infine la lista di vertici $(1, 0, 3, 4)$ rappresenta un cammino nel grafo privo di corde.



Soluzione

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

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

struct nodo *leggi_lista(void) {
  int i, n;
  struct nodo *p, *primo;

  printf("Numero di elementi della lista: ");
  scanf("%d", &n);
  printf("Inserisci gli elementi della lista: ");
  primo = NULL;
  for (i=0; i<n; i++) {
```

```
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    p->next = primo;
    primo = p;
  }
  return(primo);
}

int leggi_grafo(struct nodo *L[]) {
  int i, n;

  printf("Numero di vertici del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("Inserisci la lista dei vertici adiacenti a %d:\n", i);
    L[i] = leggi_lista();
  }
  return(n);
}

int adiacente(struct nodo *p, int x) {
  int rc;

  while (p != NULL && p->info != x)
    p = p->next;
  if (p == NULL)
    rc = 0;
  else
    rc = 1;
  return(rc);
}

int verifica_cammino(struct nodo *l, struct nodo *G[], int n) {
  struct nodo *p;
  int rc;

  rc = 1;
  while (rc && l->next != NULL) {
    if (adiacente(G[l->info], l->next->info)) {
      p = l->next->next;
      while (p != NULL && !adiacente(G[l->info], p->info))
        p = p->next;
      if (p != NULL) {
        printf("Lo spigolo (%d,%d) e' una corda in G per il cammino.\n", l->info, p->info);
        rc = 0;
      } else {
        rc = 1;
      }
    } else {
      printf("%d e %d non sono vertici adiacenti in G.\n", l->info, l->next->info);
      rc = 0;
    }
    l = l->next;
  }
  return(rc);
}
```

```
int main(void) {
  struct nodo *G[MAX], *l;
  int n;

  n = leggi_grafo(G);
  l = leggi_lista();
  if (verifica_cammino(l, G, n))
    printf("La lista rappresenta un cammino privo di corde su G.\n");
  else
    printf("La lista NON rappresenta un cammino privo di corde su G.\n");
  return(1);
}
```

Esercizio n. 2

Risolvere il seguente problema proponendo una codifica completa del programma in linguaggio C.

Costruire una lista di n numeri interi casuali compresi tra 1 e k (estremi inclusi), con n e k letti in input. Iniziando dal primo elemento della lista (x), eliminare dalla lista i successivi x elementi; quindi ripetere il procedimento dall'elemento successivo fino a quando non sarà completata l'eleborazione di tutti gli elementi rimanenti della lista. Stampare la lista degli elementi rimanenti.

Esempio Si consideri la seguente lista di elementi:

$$\mathbf{2} \rightarrow 3 \rightarrow 7 \rightarrow \mathbf{1} \rightarrow 4 \rightarrow \mathbf{3} \rightarrow 1 \rightarrow 1 \rightarrow 8 \rightarrow \mathbf{2} \rightarrow 4$$

Iniziando dal primo elemento (2) verranno cancellati gli elementi 3 e 7, quindi passando all'elemento successivo 1, verrà eliminato il 4, quindi considerando l'elemento 3 dovranno essere eliminati gli eloementi 1, 1 e 8 ed infine, esaminando l'elemento 2 sarà eliminato il 4. La lista rimanente è quindi costituita dai seguenti elementi:

$$2 \rightarrow 1 \rightarrow 3 \rightarrow 2$$

Soluzione

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

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

struct nodo *genera_lista(void) {
  struct nodo *p, *primo;
  int i, n, k;

  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Valore massimo (k): ");
  scanf("%d", &k);
  srand((unsigned)time(NULL));
  primo = NULL;
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->info = rand() % k + 1;
    p->next = primo;
```

```
    primo = p;
  }
  return(primo);
}

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

void eliminazione(struct nodo *p) {
  int i;
  struct nodo *q;

  while (p != NULL) {
    i = 0;
    while (i < p->info && p->next != NULL) {
      q = p->next;
      p->next = p->next->next;
      free(q);
      i = i+1;
    }
    p = p->next;
  }
  return;
}

int main(void) {
  struct nodo *p;

  p = genera_lista();
  printf("Lista originale: ");
  stampa_lista(p);
  eliminazione(p);
  printf("Lista residua: ");
  stampa_lista(p);
  return(1);
}
```