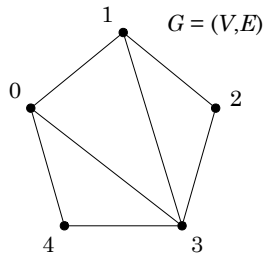
Corso di Informatica 1 (IN1 - Fondamenti) – Prof. Marco Liverani – a.a. 2005/2006

Esame scritto del 23 giugno 2006

Esercizio n. 1

Dato un grafo non orientato $G = (V, E)$, un insieme di vertici $X \subseteq V(G)$ è *dominante* per G se ogni vertice di $V(G)$ è in X o è adiacente ad almeno un vertice di X. Un insieme X dominante è *minimale* se non contiene nessun altro insieme dominante.
Letto in input un grafo non orientato $G = (V, E)$, rappresentarlo con le liste di adiacenza. Costruire una lista L con i vertici di un sottoinsieme minimale dominante di $V(G)$. Stampare la lista L.

Esempio Si consideri il grafo $G = (V, E)$ rappresentato in figura. Un insieme dominante minimale per G è $X = \{0, 2\}$; anche gli insiemi $X' = \{3\}$ e $X'' = \{2, 4\}$ sono dominanti e minimali, mentre l'insieme $Y = \{0, 1, 3\}$, pur essendo dominante non è minimale (contiene $Y' = \{3\}$ e $Y'' = \{0, 1\}$ che sono entrambi insiemi dominanti). L'insieme $Z = \{0, 4\}$ non è dominante perché il vertice 2 non è adiacente a nessun vertice in Z.



Soluzione

```
#include <stdlib.h>
#include <stdio.h>
#define MAX 50

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");
}

void stampaGrafo(struct nodo *G[], int n) {
  int i;
```

```
  for (i=0; i<n; i++) {
    printf("  %d: ", i);
    stampaLista(G[i]);
  }
  return;
}

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

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

int leggiGrafo(struct nodo *G[]) {
  int i, n;

  printf("Numero di vertici del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("Lista dei vertici adiacenti al vertice %d:\n", i);
    G[i] = leggiLista();
  }
  return(n);
}

struct nodo *aggiungi(struct nodo *p, int x) {
  struct nodo *q;

  q = malloc(sizeof(struct nodo));
  q->info = x;
  q->next = p;
  return(q);
}

int adiacenti(int x, int y, struct nodo *G[]) {
  struct nodo *p;
  int risp = 0;

  p = G[x];
  while (risp == 0 && p!= NULL && p->info != y)
    p = p->next;
  if (p != NULL)
    return(1);
  else
    return(0);
}
```

```
struct nodo *insiemeDominante(struct nodo *G[], int n) {
  struct nodo *primo = NULL, *p;
  int i;

  for (i=0; i<n; i++) {
    p = primo;
    while (p != NULL && p->info != i && !adiacenti(p->info, i, G))
      p = p->next;
    if (p == NULL)
      primo = aggiungi(primo, i);
  }
  return(primo);
}


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

  n = leggiGrafo(G);
  printf("\nListe di adiacenza del grafo:\n");
  stampaGrafo(G, n);
  L = insiemeDominante(G, n);
  printf("\nLista dei vertici dell'insieme dominante: ");
  stampaLista(L);
  return(1);
}
```

Esercizio n. 2

Letta una stringa di cifre comprese tra 0 e 9, trasformarla nel numero intero corrispondente. Stampare il numero ottenuto.

Esempio Leggendo in input la stringa $s =$ "1234" questa viene trasformata nel numero intero $n = 1234$.

Soluzione

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

int toInt(char s[]) {
  int k, p, c, n, i;

  k = strlen(s);
  n = 0;
  p = 1;
  for (i=k-1; i>=0; i--) {
    c = s[i] - 48;
    n = n + c*p;
    p = p*10;
  }
  return(n);
}
```

```
int main(void) {
  char s[10];
  int n;

  printf("Inserisci un numero: ");
  scanf("%s", s);
  n = toInt(s);
  printf("Il valore inserito e' %d.\n", n);
  return(1);
}
```