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

Esame scritto del 20 gennaio 2006

Esercizio n. 1

Letti in input tre numeri interi positivi n, m e k, tali che $0 \leq k < n$, costruire e stampare una matrice binaria A di ordine $n \times m$ di elementi casuali ($A_{i,j} \in \{0,1\}$ $\forall i = 0, \ldots, n-1$, $\forall j = 0, \ldots, m-1$). Stampare tutte le righe di A che hanno uno zero nelle colonne in cui la riga k-esima vale 1.

Esempio Siano $n = 4$, $m = 7$, $k = 2$ e si consideri la seguente matrice binaria casuale

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

La seconda e la quarta riga hanno valore 0 in tutte le colonne in cui la terza riga ($k = 2$) vale 1.

Soluzione

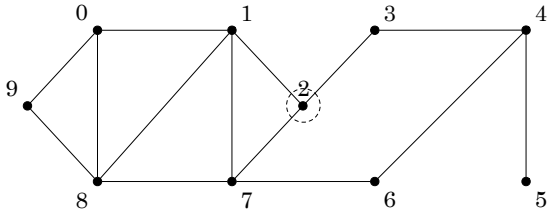
```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 100

void generaMatriceBinaria(int M[MAX][MAX], int n, int m) {
  int i, j;

  srand((unsigned)time(NULL));
  for (i=0; i<n; i++)
    for (j=0; j<m; j++)
      M[i][j] = rand()%2;
  return;
}

void stampaMatrice(int M[MAX][MAX], int n, int m) {
  int i, j;

  for (i=0; i<n; i++) {
    for (j=0; j<m; j++)
      printf("%d ", M[i][j]);
    printf("\n");
  }
  return;
}
```

```
void stampaRighe(int A[MAX][MAX], int n, int m, int k) {
  int i, j, flag;

  for (i=0; i<n; i++) {
    if (i != k) {
      flag = 0;
      for (j=0; j<m && flag == 0; j++) {
        if (A[k][j] == 1 && A[i][j] == 1)
          flag = 1;
      }
      if (flag == 0) {
        printf("Riga %d: ", i);
        for (j=0; j<m; j++)
          printf("%d ", A[i][j]);
        printf("\n");
      }
    }
  }
  return;
}

int main(void) {
  int A[MAX][MAX], n, m, k;

  printf("inserisci i tre interi positivi: ");
  scanf("%d %d %d", &n, &m, &k);
  generaMatriceBinaria(A, n, m);
  stampaMatrice(A, n, m);
  stampaRighe(A, n, m, k);
  return(1);
}
```

Esercizio n. 2

Leggere in input un grafo $G = (V, E)$ e rappresentarlo mediante liste di adiacenza. Letto in input un vertice $v \in V(G)$, costruire e stampare la lista dei vertici distinti di G, adiacenti ai vicini di v, ma non adiacenti a v.

Esempio Si consideri il grafo $G = (V, E)$ rappresentato in figura. Sia $v = 2$: allora la lista (senza ripetizioni) dei vertici adiacenti ai vicini di v, ma non adiacenti a v è la seguente: $0 \to 4 \to 6 \to 8$.



Soluzione

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

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

struct nodo *leggiLista(void) {
  struct nodo *p, *primo = NULL;
  int i, n;
  printf("  Inserisci il numero di elementi della lista: ");
  scanf("%d", &n);
  printf("  Inserisci i %d elementi della lista: ", n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    p->next = primo;
    primo = p;
  }
  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 di adiacenza del vertice %d.\n", i);
    G[i] = leggiLista();
  }
  return(n);
}

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

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 esiste(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);
}

struct nodo *listaVertici(struct nodo *G[], int n, int v) {
  struct nodo *p, *q, *primo = NULL;

  p = G[v];
  while (p != NULL) {
    q = G[p->info];
    while (q != NULL) {
      if (q->info != v && !esiste(primo, q->info) && !esiste(G[v], q->info))
        primo = aggiungi(primo, q->info);
      q = q->next;
    }
    p = p->next;
  }
  return(primo);
}

int main(void) {
  struct nodo *G[MAX], *primo = NULL;
  int n, v;

  n = leggiGrafo(G);
  printf("Inserisci il vertice v: ");
  scanf("%d", &v);
  primo = listaVertici(G, n, v);
  stampaLista(primo);
  return(1);
}
```