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

Esame scritto del 10 febbraio 2006

Esercizio n. 1

Letti in input due interi positivi n e m generare e stampare una matrice A di $n \times m$ numeri razionali casuali compresi tra 0 e 1 (si ricorda che la funzione `rand()` produce numeri casuali interi compresi tra 0 e `RAND_MAX`, costante definita nella libreria standard). Stampare gli elementi della colonna contenente il massimo numero di elementi consecutivi decrescenti.

Esempio Siano $n = 7$ e $m = 4$ e si consideri la seguente matrice casuale

$$A = \begin{pmatrix} 0,523 & 0,7 & 0,123 & 0,0009 \\ 0,47 & \mathbf{1,0} & 0,88 & 0,42 \\ 0,9012 & \mathbf{0,63} & 0,0 & 0,89 \\ 0,17 & \mathbf{0,601} & 0,54 & 0,2 \\ 0,26 & \mathbf{0,57} & 0,41 & 0,73 \\ 0,3 & \mathbf{0,407} & 0,92 & 0,03 \\ 0,68 & 0,879 & 0,1 & 0,28 \end{pmatrix}$$

La colonna con il massimo numero di elementi decrescenti consecutivi è la seconda.

Soluzione

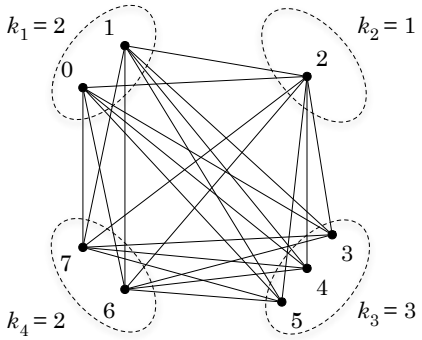
```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 50

void generaMatrice(float A[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++)
      A[i][j] = (float)rand()/RAND_MAX;
  return;
}

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

  for (i=0; i<n; i++) {
    for (j=0; j<m; j++)
      printf("%1.6f ", A[i][j]);
    printf("\n");
  }
  return;
}
```

```
void stampaColonna(float A[MAX][MAX], int n, int j) {
  int i;

  printf("Elementi della colonna %d:\n", j);
  for (i=0; i<n; i++)
    printf("%1.5f\n", A[i][j]);
  return;
}

int trovaColonna(float A[MAX][MAX], int n, int m) {
  int i, j, cont, contBis, contMax, jMax;

  contMax = 0;
  jMax = -1;

  for (j=0; j<m; j++) {
    cont = 0;
    contBis = 0;
    for (i=0; i<n-1; i++) {
      if (A[i][j]>A[i+1][j]) {
        cont++;
      } else {
        if (cont > contBis)
          contBis = cont;
        cont = 0;
      }
    }
    if (contBis > contMax) {
      contMax = contBis;
      jMax = j;
    }
  }
  return(jMax);
}

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

  printf("Numero di righe e colonne: ");
  scanf("%d %d", &n, &m);
  generaMatrice(A, n, m);
  stampaMatrice(A, n, m);
  j = trovaColonna(A, n, m);
  if (j > 0)
    stampaColonna(A, n, j);
  else
    printf("Nessuna colonna presenta elementi consecutivi decrescenti.\n");
  return(1);
}
```

Esercizio n. 2

Letti in input quattro numeri interi positivi k_1, k_2, k_3 e k_4 costruire il grafo multipartito completo con $k_1 + k_2 + k_3 + k_4$ vertici e quattro insiemi indipendenti (privi di spigoli tra i vertici di ogni insieme) di k_i vertici ciascuno ($i = 1, 2, 3, 4$). Stampare le liste di adiacenza del grafo.

Esempio Si considerino i quattro interi $k_1 = 2, k_2 = 1, k_3 = 3, k_4 = 2$. Il grafo multipartito completo ottenuto è rappresentato in figura.



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;
}

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

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

```
int generaMultipartito(struct nodo *G[], int k[]) {
  int i, j, h, n=0, m=0;

  n = k[0]+k[1]+k[2]+k[3];
  for (i=0; i<n; i++)
    G[i] = NULL;
  for (i=0; i<4; i++) {
    for (j=m; j<m+k[i]; j++) {
      for (h=m+k[i]; h<n; h++) {
        aggiungi(&G[j], h);
        aggiungi(&G[h], j);
      }
    }
    m = m+k[i];
  }
  return(n);
}

int main(void) {
  int k[4], n;
  struct nodo *G[MAX];

  printf("Inserisci quattro interi: ");
  scanf("%d %d %d %d", &k[0], &k[1], &k[2], &k[3]);
  n = generaMultipartito(G, k);
  stampaGrafo(G, n);
  return(1);
}
```