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

Esame scritto del 5 Settembre 2016 (Appello X)

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.

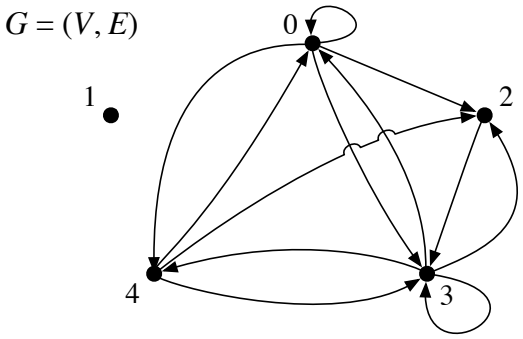
Esercizio n. 1

Letti in input due interi $n, k > 0$ generare una lista $L : x_0 \to x_1 \to x_2 \to \cdots \to x_{k-1}$ di k numeri interi positivi casuali tali che $x_i < n$ per $i = 0, 1, \ldots, k-1$. Costruire le liste di adiacenza del grafo orientato $G = (V, E)$, tale che $V = \{0, 1, \ldots, n-1\}$ ed $E = \{(i, j) : 0 \leq i, j < n$ e x_j è successivo a x_i nella lista $L\}$. Stampare la lista L e le liste di adiacenza di G.

Esempio Supponiamo che $n = 5$, $k = 6$ e che la lista di k numeri casuali minori di n sia la seguente:

$$L : 0 \to 3 \to 4 \to 0 \to 2 \to 3$$

Allora il grafo G di cui devono essere costruite le liste di adiacenza è il seguente:



Soluzione

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

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

struct nodo *costruisciLista(int n, int k) {
  struct nodo *p, *primo = NULL;
  int i;
```

```
14    srand((unsigned)time(NULL));
15    for (i=0; i<k; i++) {
16      p = malloc(sizeof(struct nodo));
17      p->next = primo;
18      p->info = rand() % n;
19      primo = p;
20    }
21    return(primo);
22  }
23
24  int adiacente(struct nodo *G[], int u, int v) {
25    struct nodo *p;
26    p = G[u];
27    while (p != NULL && p->info != v)
28      p = p->next;
29    if (p != NULL)
30      return(1);
31    else
32      return(0);
33  }
34
35  void aggiungiSpigolo(struct nodo *G[], int u, int v) {
36    struct nodo *p;
37    p = malloc(sizeof(struct nodo));
38    p->info = v;
39    p->next = G[u];
40    G[u] = p;
41    return;
42  }
43
44  void costruisciGrafo(struct nodo *G[], struct nodo *L, int n) {
45    struct nodo *p;
46    int i;
47    for (i=0; i<n; i++)
48      G[i] = NULL;
49    while (L != NULL) {
50      p = L->next;
51      while (p != NULL) {
52        if (!adiacente(G, L->info, p->info))
53          aggiungiSpigolo(G, L->info, p->info);
54        p = p->next;
55      }
56      L = L->next;
57    }
58    return;
59  }
60
61  void stampaLista(struct nodo *p) {
62    while (p != NULL) {
63      printf("%d --> ", p->info);
64      p = p->next;
65    }
66    printf("NULL\n");
67    return;
68  }
69
```

```c
void stampaGrafo(struct nodo *G[], int n) {
  int i;
  for (i=0; i<n; i++) {
    printf("%2d: ", i);
    stampaLista(G[i]);
  }
  return;
}

int main(void) {
  int n, k;
  struct nodo *L, *G[MAX];
  printf("Inserisci n e k: ");
  scanf("%d %d", &n, &k);
  L = costruisciLista(n, k);
  costruisciGrafo(G, L, n);
  printf("Lista: ");
  stampaLista(L);
  printf("Grafo:\n");
  stampaGrafo(G, n);
  return(0);
}
```

Esercizio n. 2

Letti in input due interi $n, k > 0$ con $n > k$, costruire e stampare due matrici A e B quadrate di ordine n e k rispettivamente, con valori 0 e 1 casuali. Verificare se la matrice B coincide con una sotto-matrice di A di ordine k.

Esempio Siano $n = 5$, $k = 3$ e consideriamo le seguenti matrici:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix} \quad B' = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad B'' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

La matrice B' non è una sotto-matrice di A, mentre la matrice B'' è una sotto-matrice di A (facendo coincidere $B''_{(0,0)}$ con $A_{(1,2)}$).

Soluzione

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

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

```
13    return;
14  }
15
16  void costruisciMatrice(int M[MAX][MAX], int n) {
17    int i, j;
18    for (i=0; i<n; i++)
19      for (j=0; j<n; j++)
20        M[i][j] = rand() % 2;
21    return;
22  }
23
24  int contiene(int A[MAX][MAX], int n, int B[MAX][MAX], int k) {
25    int i, j, x, y, ok=0;
26    for (i=0; i<n-k && ok == 0; i++) {
27      for (j=0; j<n-k && ok == 0; j++) {
28        ok = 1;
29        for (x=0; x<k && ok == 1; x++)
30          for (y=0; y<k && ok == 1; y++)
31            if (A[i+x][j+y] != B[x][y])
32              ok = 0;
33      }
34    }
35    return(ok);
36  }
37
38  int main(void) {
39    int A[MAX][MAX], B[MAX][MAX], n, k;
40    srand((unsigned)time(NULL));
41    printf("Inserisci l'ordine della prima e della seconda matrice: ");
42    scanf("%d %d", &n, &k);
43    costruisciMatrice(A, n);
44    costruisciMatrice(B, k);
45    stampaMatrice(A, n);
46    printf("\n");
47    stampaMatrice(B, k);
48    if (contiene(A, n, B, k))
49      printf("La matrice B e' una sottomatrice di A\n");
50    else
51      printf("La matrice B non e' una sottomatrice di A\n");
52    return(0);
53  }
```