# Algoritmi e Strutture Dati (IN110)
# **Esercitazione n. 11**

Marco Liverani *

## Esercizio n. 1

Letto in input un grafo non orientato $G = (V, E)$ con $n$ vertici ($V = \{0, 1, 2, \ldots, n-1\}$) e una lista di numeri interi compresi tra 0 e $n-1$, verificare se la lista rappresenta un cammino sul grafo.

### Codifica in linguaggio C

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

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

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo=NULL;
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d vertici: ", n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    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");
```

*Università degli Studi Roma Tre, Corso di Laurea in Matematica, Corso di Algoritmi e Strutture Dati (IN110) – sito web del corso http://www.mat.uniroma3.it/users/liverani/IN110/

```
  return;
}

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("Lista dei vertici adiacenti al vertice %d.\n", i);
    L[i] = leggi_lista();
  }
  return(n);
}

void stampa_grafo(struct nodo *L[], int n) {
  int i;
  printf("Liste di adiacenza dei vertici del grafo:\n");
  for (i=0; i<n; i++) {
    printf("%2d: ", i);
    stampa_lista(L[i]);
  }
  printf("\n");
  return;
}

int adiacente(struct nodo *G[], int u, int v) {
  struct nodo *p;
  p = G[u];
  while (p != NULL && p->info != v) {
    p = p->next;
  }
  if (p == NULL)
    return(0);
  else
    return(1);
}

int main(void) {
  struct nodo *G[100], *primo, *p;
  int n, ok=1;
  n = leggi_grafo(G);
  stampa_grafo(G, n);
  printf("Inserimento della lista di vertici da verificare\n");
  primo = leggi_lista();
  p = primo;
  while (ok && p->next != NULL) {
    if (!adiacente(G, p->info, p->next->info))
      ok = 0;
    p = p->next;
  }
  if (ok)
    printf("La lista costituisce un cammino sul grafo.\n");
  else
    printf("La lista non rappresenta un cammino sul grafo.\n");
```

```
85    return(0);
86  }
```

## Esercizio n. 2

Leggere in input $k$ liste di numeri interi e costruire un grafo non orientato $G = (V, E)$ per cui le $k$ liste rappresentino dei cammini.

### Codifica in linguaggio C

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

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

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo=NULL;
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d vertici: ", n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    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;
}

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("Lista dei vertici adiacenti al vertice %d.\n", i);
    L[i] = leggi_lista();
  }
  return(n);
}

void stampa_grafo(struct nodo *L[], int n) {
  int i;
```

```
47    printf("Liste di adiacenza dei vertici del grafo:\n");
48    for (i=0; i<n; i++) {
49      printf("%2d: ", i);
50      stampa_lista(L[i]);
51    }
52    printf("\n");
53    return;
54  }
55
56  int main(void) {
57    struct nodo *L[100], *G[100], *p, *q;
58    int i, n=0, k;
59    for (i=0; i<100; i++)
60      G[i] = NULL;
61    printf("Quante liste vuoi inserire? ");
62    scanf("%d", &k);
63    for (i=0; i<k; i++)
64      L[i] = leggi_lista();
65    for (i=0; i<k; i++) {
66      p = L[i];
67      if (n < p->info)
68        n = p->info;
69      while (p->next != NULL) {
70        if (n < p->next->info)
71          n = p->next->info;
72        q = G[p->info];
73        while (q != NULL && q->info != p->next->info) {
74          q = q->next;
75        }
76        if (q == NULL) {
77          q = malloc(sizeof(struct nodo));
78          q->info = p->next->info;
79          q->next = G[p->info];
80          G[p->info] = q;
81        }
82        p = p->next;
83      }
84    }
85    stampa_grafo(G, n+1);
86    return(0);
87  }
```

## Esercizio n. 3

Letti in input due interi $n$ e $k$ ($0 < k < n$), costruire un grafo orientato $G = (V, E)$ in modo casuale tale che $V = \{0, 1, 2, \ldots, n-1\}$ e ogni vertice abbia al massimo $k$ spigoli uscenti.

### Codifica in linguaggio C

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

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

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

void stampa_grafo(struct nodo *L[], int n) {
  int i;
  printf("Liste di adiacenza dei vertici del grafo:\n");
  for (i=0; i<n; i++) {
    printf("%2d: ", i);
    stampa_lista(L[i]);
  }
  printf("\n");
  return;
}

void crea_grafo(struct nodo *G[], int n, int k) {
  struct nodo *p;
  int i, j, x;
  srand((unsigned)time(NULL));
  for (i=0; i<n; i++) {
    G[i] = NULL;
    for (j=0; j<k; j++) {
      x = rand() % n;
      if (x != i) {
        p = G[i];
        while (p != NULL && p->info != x) {
          p = p->next;
        }
        if (p == NULL) {
          p = malloc(sizeof(struct nodo));
          p->info = x;
          p->next = G[i];
```

```
47          G[i] = p;
48        }
49      }
50    }
51  }
52  return;
53 }
54
55 int main(void) {
56   struct nodo *G[100];
57   int n, k;
58   printf("Numero di vertici: ");
59   scanf("%d", &n);
60   printf("Grado uscente massimo: ");
61   scanf("%d", &k);
62   crea_grafo(G, n, k);
63   stampa_grafo(G, n);
64   return(0);
65 }
```