Algoritmi e Strutture Dati (IN110)
Esercitazione n. 10

Marco Liverani *

Esercizio n. 1

Letto in input un grafo G orientato, rappresentarlo con liste di adiacenza. Costruire e stampare le liste di adiacenza del grafo H opposto di G: $V(H) = V(G)$, $E(H) = \{(u,v) : u,v \in V(H) \text{ e } (v,u) \in E(G)\}$ (il grafo con il verso degli spigoli invertito).

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;
  }
```

*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/

```
30    printf("NULL\n");
31    return;
32  }
33
34  int leggi_grafo(struct nodo *L[]) {
35    int i, n;
36    printf("Numero di vertici del grafo: ");
37    scanf("%d", &n);
38    for (i=0; i<n; i++) {
39      printf("Lista dei vertici adiacenti al vertice %d.\n", i);
40      L[i] = leggi_lista();
41    }
42    return(n);
43  }
44
45  void stampa_grafo(struct nodo *L[], int n) {
46    int i;
47    for (i=0; i<n; i++) {
48      printf("%2d: ", i);
49      stampa_lista(L[i]);
50    }
51    printf("\n");
52    return;
53  }
54
55  void opposto(struct nodo *G[], struct nodo *H[], int n) {
56    int i;
57    struct nodo *p, *q;
58    for (i=0; i<n; i++) {
59      H[i] = NULL;
60    }
61    for (i=0; i<n; i++) {
62      p = G[i];
63      while (p != NULL) {
64        q = malloc(sizeof(struct nodo));
65        q->info = i;
66        q->next = H[p->info];
67        H[p->info] = q;
68        p = p->next;
69      }
70    }
71    return;
72  }
73
74  int main(void) {
75    struct nodo *G[MAX], *H[MAX];
76    int n;
77    n = leggi_grafo(G);
78    stampa_grafo(G, n);
79    opposto(G, H, n);
80    stampa_grafo(H, n);
81    return(0);
82  }
```

Esercizio n. 2

Letto in input un grafo $G = (V, E)$ non orientato, costruire il grafo $H = (V, E')$ complementare, tale che $(u, v) \in E \iff (u, v) \notin E'$.

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);
}
```

```
47 void stampa_grafo(struct nodo *L[], int n) {
48   int i;
49   for (i=0; i<n; i++) {
50     printf("%2d: ", i);
51     stampa_lista(L[i]);
52   }
53   printf("\n");
54   return;
55 }
56
57 int adiacente(struct nodo *G[], int u, int v) {
58   struct nodo *p;
59   p = G[u];
60   while (p != NULL && p->info != v) {
61     p = p->next;
62   }
63   if (p == NULL)
64     return(0);
65   else
66     return(1);
67 }
68
69 void complementare(struct nodo *G[], struct nodo *H[], int n) {
70   int i, j;
71   struct nodo *p;
72   for (i=0; i<n; i++) {
73     H[i] = NULL;
74   }
75   for (i=0; i<n; i++) {
76     for (j=0; j<n; j++) {
77       if (i != j && !adiacente(G, i, j)) {
78         p = malloc(sizeof(struct nodo));
79         p->info = j;
80         p->next = H[i];
81         H[i] = p;
82       }
83     }
84   }
85   return;
86 }
87
88 int main(void) {
89   struct nodo *G[MAX], *H[MAX];
90   int n;
91   n = leggi_grafo(G);
92   stampa_grafo(G, n);
93   complementare(G, H, n);
94   stampa_grafo(H, n);
95   return(0);
96 }
```

Esercizio n. 3

Letti in input due interi n e k ($0 < k < n$), costruire un grafo $G = (V, E)$ 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 }
```