Algoritmi e Strutture Dati (IN110)
Esercitazione n. 9

Marco Liverani *

Esercizio n. 1

Letto in input un grafo orientato rappresentarlo con liste di adiacenza. Stampare le liste di adiacenza del grafo. Per ogni vertice stampare il "grado entrante" (il numero di spigoli entranti nel vertice).

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;
  int i, n;
  primo = NULL;
  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;
  }
```

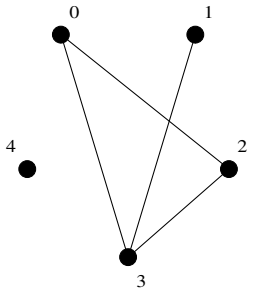
*Università degli Studi Roma Tre, Corso di Laurea in Matematica, Corso di Algoritmi e Strutture Dati (IN110) – sito web del corso http://www.mat.uniroma3.it/users/liverani/IN110/

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

Esercizio n. 2

Leggere in input un insieme di n intervalli della retta reale $I_0 = (x_0, x_1), I_1 = (x_2, x_3), \ldots, I_{n-1} = (x_{2(n-1)}, x_{2(n-1)+1})$. Utilizzando le liste di adiacenza costruire il grafo $G = (V, E)$ con n vertici ottenuto per intersezione degli intervalli I_k, $k = 0, \ldots, n-1$, nel seguente modo: ogni intervallo I_k corrisponde ad un vertice $v_k \in V(G)$; due vertici sono adiacenti se e solo se i rispettivi intervalli si intersecano: $(v_i, v_j) \in E(G) \iff I_i \cap I_j \neq \emptyset$. Stampare le liste di adiacenza di G.

Esempio Consideriamo la seguente famiglia di $n = 5$ intervalli: $I_0 = (-2, 7), I_1 = (-8, -3), I_2 = (1, 2), I_3 = (-4, 2), I_4 = (8, 9)$. Il grafo intersezione ottenuto è rappresentato in figura:



Codifica in linguaggio C

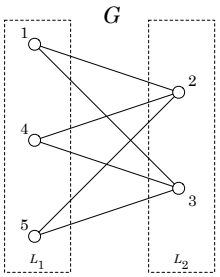
```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 100

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

struct intervallo {
  float x, y;
};

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 *G[], int n) {
  int i;
  for (i=0; i<n; i++) {
    printf("%d: ", i);
    stampa_lista(G[i]);
  }
```

```
  return;
}

int leggi_intervalli(struct intervallo A[]) {
  int i, n;
  printf("Numero di intervalli: ");
  scanf("%d", &n);
  printf("Inserisci %d coppie: ", n);
  for (i=0; i<n; i++)
    scanf("%f %f", &A[i].x, &A[i].y);
  return(n);
}

void grafo_intervallo(struct nodo *G[], struct intervallo I[], int n) {
  int i, j;
  struct nodo *p;
  for (i=0; i<n; i++)
    G[i] = NULL;
  for (i=0; i<n-1; i++) {
    for (j=i+1; j<n; j++) {
      if (I[i].x < I[j].y && I[i].y > I[j].x) {
        p = malloc(sizeof(struct nodo));
        p->info = j;
        p->next = G[i];
        G[i] = p;
        p = malloc(sizeof(struct nodo));
        p->info = i;
        p->next = G[j];
        G[j] = p;
      }
    }
  }
  return;
}

int main(void) {
  struct nodo *G[MAX];
  struct intervallo I[MAX];
  int n;
  n = leggi_intervalli(I);
  grafo_intervallo(G, I, n);
  stampa_grafo(G, n);
  return(0);
}
```

Esercizio n. 3

Leggere in input due liste di numeri interi L_1 e L_2 tali che costituiscano una partizione dell'insieme $\{1,2,\ldots,n\}$. Costruire il grafo bipartito completo G ottenuto sommando L_1 e L_2: $G = (V,E)$ dove $V(G) = L_1 \cup L_2$ e $E(G) = \{(u,v) : u \in L_1 \text{ e } v \in L_2 \text{ o viceversa}\}$. Stampare le liste di adiacenza di G.

Esempio Sia $L_1 = (1,4,5)$ e $L_2 = (2,3)$; in questo caso dunque $n = 5$ infatti $L_1 \cup L_2 = \{1,2,3,4,5\}$ e $L_1 \cap L_2 = \emptyset$. Dunque il grafo $G = (V,E)$ ottenuto come somma di L_1 e L_2 è dato da $V(G) = \{1,2,3,4,5\}$ e $E(G) = \{(1,2),(1,3),(4,2),(4,3),(5,2),(5,3)\}$, intendendo ogni coppia come non ordinata.



Codifica in linguaggio C

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

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 gli %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);
}

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

```
31    return;
32  }
33
34  void stampa_grafo(struct nodo *G[], int n) {
35    int i;
36    for (i=1; i<=n; i++) {
37      printf("N(%d) = ", i);
38      stampa_lista(G[i]);
39    }
40    return;
41  }
42
43  int costruisci_grafo(struct nodo *a, struct nodo *b, struct nodo *c[]) {
44    int i, n = 0;
45    struct nodo *pa, *pb, *pc;
46    pa = a;
47    while (pa != NULL) {
48      if (n < pa->info)
49        n = pa->info;
50      pa = pa->next;
51    }
52    pb = b;
53    while (pb != NULL) {
54      if (n < pb->info)
55        n = pb->info;
56      pb = pb->next;
57    }
58    for (i=0; i<=n; i++) {
59      c[i] = NULL;
60    }
61
62    pa = a;
63    while (pa != NULL) {
64      pb = b;
65      while (pb != NULL) {
66        pc = malloc(sizeof(struct nodo));
67        pc->info = pb->info;
68        pc->next = c[pa->info];
69        c[pa->info] = pc;
70
71        pc = malloc(sizeof(struct nodo));
72        pc->info = pa->info;
73        pc->next = c[pb->info];
74        c[pb->info] = pc;
75        pb = pb->next;
76      }
77      pa = pa->next;
78    }
79    return(n);
80  }
81
82  int main(void) {
83    struct nodo *L1, *L2, *G[MAX];
84    int n;
```

```
85    L1 = leggi_lista();
86    L2 = leggi_lista();
87    n = costruisci_grafo(L1, L2, G);
88    stampa_grafo(G, n);
89    return(0);
90  }
```