Esercizi su liste e grafi

Marco Liverani*

Gennaio 2011

*E-mail: liverani@mat.uniroma3.it – Web: http://www.mat.uniroma3.it/users/liverani

1 Esercizi sulle liste

Esercizio 1 *Letta in input una sequenza di numeri interi positivi memorizzarla in una lista. Costruire una seconda lista contenente soltanto gli elementi della prima lista che non siano numeri primi. Stampare la seconda lista.*

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

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

int primo(int x) {
  int i, flag;
  flag = 1;
  i = 2;
  while (i<=sqrt(x) && flag==1) {
    if (x%i == 0)
      flag = 0;
    i = i+1;
  }
  return(flag);
}

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo;
  int i, n;
  primo = NULL;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("inserisci %d numeri interi positivi: ", 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("\n");
  return;
}

int main(void) {
  struct nodo *p, *q, *r;
```

```
49    p = leggi_lista();
50    r = NULL;
51    while (p != NULL) {
52      if (primo(p->info) == 0) {
53        q = malloc(sizeof(struct nodo));
54        q->info = p->info;
55        q->next = r;
56        r = q;
57      }
58      p = p->next;
59    }
60    stampa_lista(r);
61    return(0);
62  }
```

Esercizio 2 *Letta in input una lista, costruire una seconda lista con gli elementi disposti al contrario rispetto alla lista originale (es.: se la lista letta in input è* $1 \to 8 \to 4$ *viene costruita la lista* $4 \to 8 \to 1$*). Stampare la seconda lista.*

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

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 numeri: ", 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;
}
struct nodo *inverti(struct nodo *p) {
  struct nodo *q, *q1 = NULL;
  while (p != NULL) {
    q = malloc(sizeof(struct nodo));
    q->info = p->info;
    q->next = q1;
    q1 = q;
    p = p->next;
  }
  return(q1);
}
int main(void) {
  struct nodo *p, *q;
  p = leggi_lista();
  q = inverti(p);
  stampa_lista(q);
  return(0);
}
```

Esercizio 3 *Lette in input due liste di numeri interi ognuna delle quali ordinata, costruire una terza lista di numeri interi ordinata, ottenuta mediante la "fusione" delle prime due. Stampare la lista.*

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

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

struct nodo *fusione(struct nodo *p1, struct nodo *p2) {
  struct nodo *p, *primo = NULL;
  while (p1 != NULL && p2 != NULL) {
    p = malloc(sizeof(struct nodo));
    p->next = primo;
    primo = p;
    if (p1->info > p2->info) {
      p->info = p1->info;
      p1 = p1->next;
    } else {
      p->info = p2->info;
      p2 = p2->next;
    }
  }
  while (p1 != NULL) {
    p = malloc(sizeof(struct nodo));
    p->info = p1->info;
    p->next = primo;
    primo = p;
    p1 = p1->next;
  }
  while (p2 != NULL) {
    p = malloc(sizeof(struct nodo));
    p->info = p2->info;
    p->next = primo;
    primo = p;
    p2 = p2->next;
  }
  return(primo);
}

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

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

```
51    printf("Numero di elementi: ");
52    scanf("%d", &n);
53    printf("Inserisci %d numeri: ", n);
54    for (i=0; i<n; i++) {
55      p = malloc(sizeof(struct nodo));
56      scanf("%d", &p->info);
57      p->next = primo;
58      primo = p;
59    }
60    return(primo);
61  }
62
63  int main(void) {
64    struct nodo *p, *q, *r;
65    p = leggi_lista();
66    q = leggi_lista();
67    r = fusione(p, q);
68    stampa_lista(r);
69    return(0);
70  }
```

Esercizio 4 *Letta in input una lista di numeri interi stamparla e poi ordinarla utilizzando l'algoritmo* SELECTION SORT. *Stampare la lista ordinata.*

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

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

void scambia(int *a, int *b) {
  int c;
  c = *a;
  *a = *b;
  *b = c;
  return;
}

void selection_sort(struct nodo *p) {
  struct nodo *q;
  while (p != NULL) {
    q = p->next;
    while (q != NULL) {
      if (p->info > q->info)
        scambia(&p->info, &q->info);
      q = q->next;
    }
    p = p->next;
  }
  return;
}

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo=NULL;
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d numeri: ", 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");
```

```
52 }
53
54 int main(void) {
55   struct nodo *p;
56   p = leggi_lista();
57   stampa_lista(p);
58   selection_sort(p);
59   stampa_lista(p);
60   return(0);
61 }
```

Esercizio 5 *Leggere in input una sequenza di n numeri* floating point *e rappresentarla con una lista L; stampare la lista. Costruire una seconda lista L' composta dai soli elementi di L maggiori della media; stampare L'.*

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

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

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

void stampa_lista(struct nodo *p) {
  while (p != NULL) {
    printf("%3.2f --> ", p->info);
    p = p->next;
  }
  printf("Null\n");
  return;
}

float media(struct nodo *p) {
  float s=0.0, n=0.0;
  while (p != NULL) {
    s = s + p->info;
    n = n + 1;
    p = p->next;
  }
  return(s/n);
}

int main(void) {
  struct nodo *p, *q, *r, *s;
  float m;
  p = leggi_lista();
  stampa_lista(p);
  m = media(p);
  r = NULL;
```

```
51    q = p;
52    while (q != NULL) {
53      if (q->info > m) {
54        s = malloc(sizeof(struct nodo));
55        s->info = q->info;
56        s->next = r;
57        r = s;
58      }
59      q = q->next;
60    }
61    stampa_lista(r);
62    return(0);
63  }
```

Esercizio 6 *Leggere in input una sequenza di numeri interi ordinati in ordine crescente. Dopo aver memorizzato la sequenza in una lista, inserire nella posizione corretta all'interno della lista, tutti i numeri mancanti. Stampare in output la lista. Non devono essere usate altre liste o array di appoggio.*

Esempio Supponiamo che sia fornita in input la sequenza $4,7,8,9,15,17,21$. Dopo aver memorizzato gli elementi nella lista $4 \to 7 \to 8 \to \ldots \to 21$, vengono inseriti i numeri mancanti, ottenendo la lista composta dagli elementi $4 \to 5 \to 6 \to 7 \to 8 \to \ldots \to 19 \to 20 \to 21$.

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

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 numeri interi in ordine crescente: ", n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    p->next = primo;
    primo = p;
  }
  return(primo);
}

void completa_lista(struct nodo *p) {
  struct nodo *q;
  while (p->next != NULL) {
    if (p->info > p->next->info + 1) {
      q = malloc(sizeof(struct nodo));
      q->info = p->next->info + 1;
      q->next = p->next;
      p->next = q;
    } else
      p = p->next;
  }
  return;
}

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

```
45 }
46
47 int main(void) {
48   struct nodo *primo;
49   primo = leggi_lista();
50   completa_lista(primo);
51   stampa_lista(primo);
52   return(0);
53 }
```

2 Esercizi sui grafi

Esercizio 7 *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: $V(H) = V(G)$, $E(H) = \{(u,v) : u,v \in V(H) \ e \ (v,u) \in E(G)\}$ (il grafo con il verso degli spigoli invertito).*

```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;
  for (i=0; i<n; i++) {
    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 8 *Letto in input un grafo $G = (V, E)$ non orientato, costruire il grafo $H = (V, E')$ complementare, tale che $(u, v) \in E \iff (u, v) \notin E'$.*

```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;
  for (i=0; i<n; i++) {
    printf("%2d: ", i);
    stampa_lista(L[i]);
  }
  printf("\n");
```

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

Esercizio 9 *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,...,n−1} e ogni vertice abbia al massimo k spigoli uscenti.*

```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];
          G[i] = p;
        }
      }
    }
```

```
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  }
```

Esercizio 10 *Letto in input un grafo non orientato $G = (V, E)$ con n vertici ($V = \{0, 1, \ldots, n-1\}$) e una lista di numeri interi compresi tra 0 e $n-1$, verificare se la lista rappresenta un cammino sul grafo.*

```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;
  }
  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;
  printf("Liste di adiacenza dei vertici del grafo:\n");
  for (i=0; i<n; i++) {
    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  int main(void) {
70    struct nodo *G[100], *primo, *p;
71    int n, ok=1;
72    n = leggi_grafo(G);
73    stampa_grafo(G, n);
74    printf("Inserimento della lista di vertici da verificare\n");
75    primo = leggi_lista();
76    p = primo;
77    while (ok && p->next != NULL) {
78      if (!adiacente(G, p->info, p->next->info))
79        ok = 0;
80      p = p->next;
81    }
82    if (ok)
83      printf("La lista costituisce un cammino sul grafo.\n");
84    else
85      printf("La lista non rappresenta un cammino sul grafo.\n");
86    return(1);
87  }
```

Esercizio 11 *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.*

```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;
  }
  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;
  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 main(void) {
  struct nodo *L[100], *G[100], *p, *q;
  int i, n=0, k;
  for (i=0; i<100; i++)
    G[i] = NULL;
  printf("Quante liste vuoi inserire? ");
  scanf("%d", &k);
  for (i=0; i<k; i++)
    L[i] = leggi_lista();
  for (i=0; i<k; i++) {
    p = L[i];
    if (n < p->info)
      n = p->info;
    while (p->next != NULL) {
      if (n < p->next->info)
        n = p->next->info;
      q = G[p->info];
      while (q != NULL && q->info != p->next->info) {
        q = q->next;
      }
      if (q == NULL) {
        q = malloc(sizeof(struct nodo));
        q->info = p->next->info;
        q->next = G[p->info];
        G[p->info] = q;
      }
      p = p->next;
    }
  }
  stampa_grafo(G, n+1);
  return(0);
}
```

Esercizio 12 *Leggere in input un grafo orientato $G = (V, E)$ e rappresentarlo mediante liste di adiacenza. Stampare le liste di adiacenza del grafo. Letto in input un vertice $v \in V(G)$ $(0 \leq v \leq n)$ stampare tutti i vertici $u \in V(G)$ a cui v è adiacente (tutti i vertici u tali che $(u, v) \in E(G)$).*

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

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

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo;
  int i, n;
  printf(" inserisci il numero di elementi: ");
  scanf("%d", &n);
  printf(" inserisci %d elementi: ", n);
  primo = NULL;
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->next = primo;
    scanf("%d", &p->info);
    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 *G[]) {
  int i, n;
  printf("Inserisci il numero di vertici del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("Lista di adiacenza del vertice %d:\n", i);
    G[i] = leggi_lista();
  }
  return(n);
}

void stampa_grafo(struct nodo *G[], int n) {
  int i;
  printf("Liste di adiacenza dei vertici del grafo:\n");
  for (i=0; i<n; i++) {
    printf(" vertici adiacenti a %d: ", i);
```

```
50      stampa_lista(G[i]);
51    }
52    return;
53  }
54
55  int adiacente(int a, int b, struct nodo *G[]) {
56    struct nodo *p;
57    int r;
58    p = G[a];
59    while (p != NULL && p->info != b)
60      p = p->next;
61    if (p == NULL)
62      r = 0;
63    else
64      r = 1;
65    return(r);
66  }
67
68  int main(void) {
69    struct nodo *L[20];
70    int n, u, v;
71    n = leggi_grafo(L);
72    stampa_grafo(L, n);
73    printf("Inserisci un vertice del grafo: ");
74    scanf("%d", &v);
75    printf("Vertici a cui %d e' adiacente: ", v);
76    for (u=0; u<n; u++)
77      if (adiacente(u, v, L))
78        printf("%d ", u);
79    printf("\n");
80    return(0);
81  }
```

Esercizio 13 *Leggere in input un grafo orientato $G = (V, E)$ e rappresentarlo mediante liste di adiacenza. Stampare le liste di adiacenza del grafo. Letto in input un vertice v $(0 \leq v \leq n)$ eliminare tutti gli spigoli entranti in v. Stampare le liste di adiacenza del grafo.*

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

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

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo;
  int i, n;
  printf(" inserisci il numero di elementi: ");
  scanf("%d", &n);
  printf(" inserisci %d elementi: ", n);
  primo = NULL;
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->next = primo;
    scanf("%d", &p->info);
    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 *G[]) {
  int i, n;
  printf("Inserisci il numero di vertici del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("Lista di adiacenza del vertice %d:\n", i);
    G[i] = leggi_lista();
  }
  return(n);
}

void stampa_grafo(struct nodo *G[], int n) {
  int i;
  printf("Liste di adiacenza dei vertici del grafo:\n");
  for (i=0; i<n; i++) {
    printf(" vertici adiacenti a %d: ", i);
```

```
    stampa_lista(G[i]);
  }
  return;
}

void elimina_spigoli(struct nodo *G[], int n, int v) {
  int i;
  struct nodo *p, *prec;
  for (i=0; i<n; i++) {
    p = G[i];
    prec = NULL;
    while (p != NULL && p->info != v) {
      prec = p;
      p = p->next;
    }
    if (p != NULL) {
      if (prec == NULL) {
        G[i] = p->next;
        free(p);
      } else {
        prec->next = p->next;
        free(p);
      }
    }
  }
  return;
}

int main(void) {
  struct nodo *L[20];
  int n, v;
  n = leggi_grafo(L);
  stampa_grafo(L, n);
  printf("Inserisci un vertice del grafo: ");
  scanf("%d", &v);
  elimina_spigoli(L, n, v);
  stampa_grafo(L, n);
  return(0);
}
```

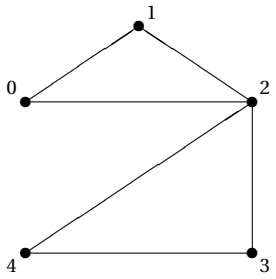
Esercizio 14 *Letto un grafo non orientato $G = (V, E)$ e letta una lista di vertici di V, $L = \{v_1, \ldots, v_k\}$, stabilire se il sottografo G' indotto da L è completo. Un sottografo è* completo *se i suoi vertici sono adiacenti a tutti gli altri vertici del sottografo. Il grafo G' è il* sottografo indotto *di G mediante l'insieme di vertici $L \subseteq V(G)$ se gli spigoli di G' sono tutti e soli gli spigoli di G aventi per estremi vertici in L.*

```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(" inserisci il numero di elementi: ");
  scanf("%d", &n);
  printf(" inserisci %d elementi: ", n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->next = primo;
    scanf("%d", &p->info);
    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 *G[]) {
  int i, n;
  printf("Inserisci il numero di vertici del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("Lista di adiacenza del vertice %d:\n", i);
    G[i] = leggi_lista();
  }
  return(n);
}

void stampa_grafo(struct nodo *G[], int n) {
  int i;
  printf("Liste di adiacenza dei vertici del grafo:\n");
  for (i=0; i<n; i++) {
```

```
49      printf(" vertici adiacenti a %d: ", i);
50      stampa_lista(G[i]);
51    }
52    return;
53  }
54
55  int adiacente(int i, int j, struct nodo *G[]) {
56    struct nodo *p;
57    p = G[i];
58    while (p != NULL && p->info != j) {
59      p = p->next;
60    }
61    if (p == NULL)
62      return(0);
63    else
64      return(1);
65  }
66
67  int completo(struct nodo *G[], struct nodo *L, int n) {
68    struct nodo *p, *q;
69    int flag=1;
70    p = L;
71    while (p != NULL && flag==1) {
72      q = L;
73      while (q != NULL && flag==1) {
74        if (q->info != p->info && !adiacente(p->info, q->info, G))
75          flag = 0;
76        q = q->next;
77      }
78      p = p->next;
79    }
80    return(flag);
81  }
82
83  int main(void) {
84    struct nodo *G[MAX], *L;
85    int n;
86    n = leggi_grafo(G);
87    stampa_grafo(G, n);
88    L = leggi_lista();
89    if (completo(G, L, n)) {
90      printf("Il sottografo di G indotto da L e‘ completo.\n");
91    } else {
92      printf("Il sottografo di G indotto da L NON e‘ completo.\n");
93    }
94    return(0);
95  }
```

Esercizio 15 *Letto un grafo non orientato $G = (V, E)$ rappresentarlo con liste di adiacenza. Letta in input un sottoinsieme di vertici di V, rappresentarla mediante una lista L ($L \subseteq V$). Verificare se L è una* copertura di vertici *di G, ossia se per ogni $(u, v) \in E$ risulta $u \in L$ o $v \in L$ (o entrambi).*

Esempio Consideriamo il grafo $G = (V, E)$ rappresentato in figura, in cui l'insieme dei vertici è $V = \{0, 1, 2, 3, 4\}$ e l'insieme degli spigoli è $E = \{(0,1), (0,2), (1,2), (2,3), (2,4), (3,4)\}$:



La lista $L = \{1, 2, 3\}$ è una copertura di G, mentre $L' = \{0, 1, 3\}$ non è una copertura, infatti gli estremi dello spigolo $(4, 2)$ non appartengono a L'.

```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 20

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

struct nodo *leggi_lista(void) {
  int i, n;
  struct nodo *p, *primo=NULL;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d elementi: ", 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;
}

```

```
34 int leggi_grafo(struct nodo *v[]) {
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 di adiacenza del vertice %d\n", i);
40     v[i] = leggi_lista();
41   }
42   return(n);
43 }
44
45 void stampa_grafo(struct nodo *v[], int n) {
46   int i;
47   printf("Liste di adiacenza del grafo\n");
48   for (i=0; i<n; i++) {
49     printf("%d: ", i);
50     stampa_lista(v[i]);
51   }
52   return;
53 }
54
55 int appartiene(int i, struct nodo *t) {
56   int trovato = 0;
57   while (t!=NULL && !trovato) {
58     if (t->info == i)
59       trovato = 1;
60     t = t->next;
61   }
62   return(trovato);
63 }
64
65 int copertura(struct nodo *v[], int n, struct nodo *p) {
66   struct nodo *q;
67   int i, flag=1;
68   for (i=0; i<n && flag; i++) {
69     flag = 0;
70     if (!appartiene(i, p)) {
71       q = v[i];
72       flag = 1;
73       while (flag && q!=NULL) {
74         if (!appartiene(q->info, p))
75           flag = 0;
76         q = q->next;
77       }
78     } else
79       flag = 1;
80   }
81   return(flag);
82 }
83
84 int main(void) {
85   struct nodo *v[MAX], *p;
86   int n;
87   n = leggi_grafo(v);
```

```
88    p = leggi_lista();
89    if (copertura(v, n, p))
90      printf("La lista e‘ una copertura del grafo.\n");
91    else
92      printf("La lista non e‘ una copertura del grafo.\n");
93    return(0);
94  }
```

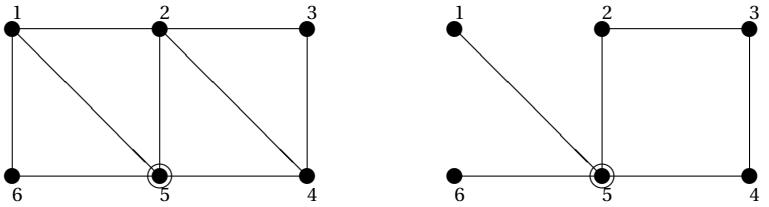
Esercizio 16 *Letto in input un grafo $G = (V,E)$ non orientato, costruire e stampare un grafo $G' = (V,E')$ orientato, tale che $(u,v) \in E'$ se e solo se $g(u) > g(v)$, dove con $g(v)$ si è indicato il grado del vertice v (il numero di spigoli entranti o uscenti: in un grafo non orientato non c'è distinzione).*

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

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

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo;
  int i, n;
  printf(" inserisci il numero di elementi: ");
  scanf("%d", &n);
  printf(" inserisci %d elementi: ", n);
  primo = NULL;
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->next = primo;
    scanf("%d", &p->info);
    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 *G[]) {
  int i, n;
  printf("Inserisci il numero di vertici del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("Lista di adiacenza del vertice %d:\n", i);
    G[i] = leggi_lista();
  }
  return(n);
}

void stampa_grafo(struct nodo *G[], int n) {
  int i;
  printf("Liste di adiacenza dei vertici del grafo:\n");
  for (i=0; i<n; i++) {
    printf(" vertici adiacenti a %d: ", i);
```

```
50      stampa_lista(G[i]);
51    }
52    return;
53  }
54
55  int grado(struct nodo *p) {
56    int cont = 0;
57    while (p!=NULL) {
58      cont ++;
59      p = p->next;
60    }
61    return(cont);
62  }
63
64  void costruisci_grafo(struct nodo *G[], struct nodo *G1[], int n) {
65    int i, gi;
66    struct nodo *p, *q;
67    for (i=0; i<n; i++) {
68      gi = grado(G[i]);
69      p = G[i];
70      G1[i] = NULL;
71      while (p != NULL) {
72        if (gi > grado(G[p->info])) {
73          q = malloc(sizeof(struct nodo));
74          q->info = p->info;
75          q->next = G1[i];
76          G1[i] = q;
77        }
78        p = p->next;
79      }
80    }
81    return;
82  }
83
84  int main(void) {
85    struct nodo *G[100], *G1[100];
86    int n;
87    n = leggi_grafo(G);
88    stampa_grafo(G, n);
89    costruisci_grafo(G, G1, n);
90    stampa_grafo(G1, n);
91    return(0);
92  }
```

Esercizio 17 *Leggere in input un grafo $G = (V, E)$ non orientato e memorizzarlo mediante liste di adiacenza. Scelto arbitrariamente uno dei vertici $v \in V$ di grado massimo, eliminare dal grafo tutti gli spigoli $(u, w) \in E$ per ogni u e w adiacenti a v. Stampare le liste di adiacenza del grafo così modificato.*

Esempio Sia $G = (V, E)$ il grafo letto in input rappresentato in figura (a sinistra), con $V = \{1,2,3,4,5,6\}$ ed $E = \{(1,2),(2,3),(3,4),(4,5),(5,6),(6,1),(1,5),(2,5),(2,4)\}$. I vertici di grado massimo sono 2 e 5 (entrambi di grado 4). Scegliendo il vertice 5, devono essere eliminati gli spigoli $(1,2)$ (perché $1,2 \in N(5)$), $(1,6)$ (perché $1,6 \in N(5)$) e $(2,4)$ (perché $4,2 \in N(5)$). si ottiene così il grafo rappresentato a destra nella figura.



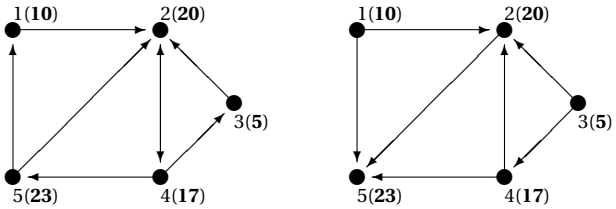

```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 30

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

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

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

```

```
35 int leggi_grafo(struct nodo *G[]) {
36   int i, n;
37   printf("Inserisci il numero di vertici del grafo: ");
38   scanf("%d", &n);
39   for (i=0; i<n; i++) {
40     printf("Lista di adiacenza del vertice %d:\n", i);
41     G[i] = leggi_lista();
42   }
43   return(n);
44 }
45
46 void stampa_grafo(struct nodo *G[], int n) {
47   int i;
48   printf("Liste di adiacenza dei vertici del grafo:\n");
49   for (i=0; i<n; i++) {
50     printf(" vertici adiacenti a %d: ", i);
51     stampa_lista(G[i]);
52   }
53   return;
54 }
55
56 int grado(struct nodo *p) {
57   int cont = 0;
58   while (p!=NULL) {
59     cont ++;
60     p = p->next;
61   }
62   return(cont);
63 }
64
65 struct nodo *elimina(int v, struct nodo *primo) {
66   struct nodo *p, *q = NULL;
67   if (primo != NULL) {
68     if (primo->info == v) {
69       q = primo;
70       primo = primo->next;
71     } else {
72       p = primo;
73       while (p->next != NULL && p->next->info != v) {
74         p = p->next;
75       }
76       if (p->next != NULL) {
77         q = p->next;
78         p->next = p->next->next;
79       }
80     }
81     if (q != NULL)
82       free(q);
83   }
84   return(primo);
85 }
86
87
88 int main(void) {
```

```
  struct nodo *G[MAX], *p, *q;
  int n, gmax, vmax, v, g;
  n = leggi_grafo(G);
  gmax = grado(G[0]);
  vmax = 0;
  for (v=1; v<n; v++) {
    g = grado(G[v]);
    if (g > gmax) {
      gmax = g;
      vmax = v;
    }
  }
  printf("Il vertice di grado massimo scelto e' %d.\n", vmax);
  p = G[vmax];
  while (p->next != NULL) {
    q = p->next;
    while (q != NULL) {
      G[q->info] = elimina(p->info, G[q->info]);
      G[p->info] = elimina(q->info, G[p->info]);
      q = q->next;
    }
    p = p->next;
  }
  stampa_grafo(G, n);
  return(0);
}
```

Esercizio 18 *Leggere in input un grafo orientato $G = (V,E)$ e rappresentarlo mediante liste di adiacenza. Leggere in input un insieme di pesi (interi) associati ai vertici del grafo: $\{w_1,...,w_n\}$. Modificando le liste di adiacenza con cui è stato rappresentato il grafo G, variare l'orientamento degli spigoli in modo tale che per ogni spigolo (u,v) risulti $w_u \leq w_v$.*

Esempio Sia $G = (V,E)$ il grafo orientato letto in input rappresentato in figura, con $V = \{1,2,3,4,5\}$ ed $E = \{(1,2),(2,4),(3,2),(4,2),(4,3),(4,5),(5,1),(5,2)\}$. Sia W l'insieme dei pesi associati ai vertici del grafo: $W = \{10,30,5,17,23\}$. Sulla sinistra è rappresentato il grafo letto in input e sulla destra il grafo prodotto dalla rielaborazione richiesta dall'esercizio.




```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;
  int i, n;
  printf(" inserisci il numero di elementi: ");
  scanf("%d", &n);
  printf(" inserisci %d elementi: ", n);
  primo = NULL;
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->next = primo;
    scanf("%d", &p->info);
    primo = p;
  }
  return(primo);
}

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

```
34
35  int leggi_grafo(struct nodo *G[]) {
36    int i, n;
37    printf("Inserisci il numero di vertici del grafo: ");
38    scanf("%d", &n);
39    for (i=0; i<n; i++) {
40      printf("Lista di adiacenza del vertice %d:\n", i);
41      G[i] = leggi_lista();
42    }
43    return(n);
44  }
45
46  void stampa_grafo(struct nodo *G[], int n) {
47    int i;
48    printf("Liste di adiacenza dei vertici del grafo:\n");
49    for (i=0; i<n; i++) {
50      printf(" vertici adiacenti a %d: ", i);
51      stampa_lista(G[i]);
52    }
53    return;
54  }
55
56  void leggi_pesi(int w[], int n) {
57    int i;
58    printf("Inserisci i pesi assegnati ai vertici del grafo:\n");
59    for (i=0; i<n; i++) {
60      printf(" w(%d) = ", i);
61      scanf("%d", &w[i]);
62    }
63    return;
64  }
65
66  void aggiungi(struct nodo *G[], int i, int j) {
67    struct nodo *p;
68    p = G[i];
69    while (p != NULL && p->info != j)
70      p = p->next;
71    if (p == NULL) {
72      p = malloc(sizeof(struct nodo));
73      p->info = j;
74      p->next = G[i];
75      G[i] = p;
76    }
77    return;
78  }
79
80  int main(void) {
81    struct nodo *G[MAX], *p, *prec;
82    int i, n, w[MAX];
83    n = leggi_grafo(G);
84    leggi_pesi(w, n);
85    for (i=0; i<n; i++) {
86      p = G[i];
87      prec = NULL;
```

```
    while (p != NULL) {
      if (w[i] > w[p->info]) {
        aggiungi(G, p->info, i);
        if (prec != NULL) {
          prec->next = p->next;
          free(p);
          p = prec->next;
        } else {
          G[i] = p->next;
          free(p);
          p = G[i];
        }
      } else {
        prec = p;
        p = p->next;
      }
    }
  }
  stampa_grafo(G, n);
  return(0);
}
```