Corso di Algoritmi e Strutture Dati (IN110)
Tutorato n. 11

Marco Liverani*

Esercizio n. 1

Leggere in input una lista di numeri interi ordinati in ordine crescente. Dopo aver letto la sequenza, 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, vengono inseriti i numeri mancanti, ottenendo la lista composta dagli elementi $4,5,6,7,8,\ldots,18,19,20,21$.

Codifica in linguaggio C

```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 stampa_lista(struct nodo *p) {
  while (p != NULL) {
```

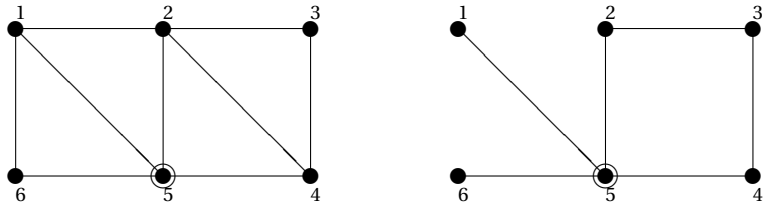
*Università degli Studi Roma Tre, Corso di Laurea in Matematica, Corso di Algoritmi e Strutture Dati (IN110); e-mail liverani@mat.uniroma3.it – sito web del corso http://www.mat.uniroma3.it/users/liverani/IN110/

```
    printf("%d --> ", p->info);
    p = p->next;
  }
  printf("Null\n");
  return;
}

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;
}

int main(void) {
  struct nodo *primo;
  primo = leggi_lista();
  completa_lista(primo);
  stampa_lista(primo);
  return(0);
}
```

Esercizio n. 2

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.



Codifica in linguaggio C

```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;
  }
```

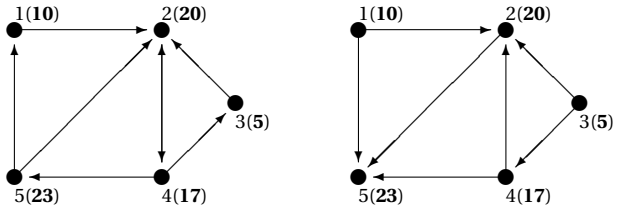
```
  printf("Null\n");
  return;
}

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;
}

int grado(struct nodo *p) {
  int cont = 0;
  while (p!=NULL) {
    cont ++;
    p = p->next;
  }
  return(cont);
}

struct nodo *elimina(int v, struct nodo *primo) {
  struct nodo *p, *q = NULL;
  if (primo != NULL) {
    if (primo->info == v) {
      q = primo;
      primo = primo->next;
    } else {
      p = primo;
      while (p->next != NULL && p->next->info != v) {
        p = p->next;
      }
      if (p->next != NULL) {
        q = p->next;
        p->next = p->next->next;
      }
    }
    if (q != NULL)
      free(q);
  }
  return(primo);
```

```
}

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 n. 3

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.



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;
  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;
  }
```

```
31    printf("Null\n");
32    return;
33  }
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);
```

```
  for (i=0; i<n; i++) {
    p = G[i];
    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);
}
```