Corso di Algoritmi e Strutture Dati (IN110)
Tutorato n. 10

Marco Liverani*

Esercizio n. 1

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.

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(" 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;
```

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

int completo(struct nodo *G[], struct nodo *L, int n) {
  struct nodo *p, *q;
  int flag=1;
  p = L;
  while (p != NULL && flag==1) {
    q = L;
    while (q != NULL && flag==1) {
      if (q->info != p->info && !adiacente(p->info, q->info, G))
        flag = 0;
      q = q->next;
    }
    p = p->next;
  }
  return(flag);
}

```

```
int main(void) {
  struct nodo *G[MAX], *L;
  int n;
  n = leggi_grafo(G);
  stampa_grafo(G, n);
  L = leggi_lista();
  if (completo(G, L, n)) {
    printf("Il sottografo di G indotto da L e' completo.\n");
  } else {
    printf("Il sottografo di G indotto da L NON e' completo.\n");
  }
  return(0);
}
```

Esercizio n. 2

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 $4 \to 5 \to 6 \to 7 \to 8 \to \ldots \to 19 \to 20 \to 21$.

Codifica in linguaggio C

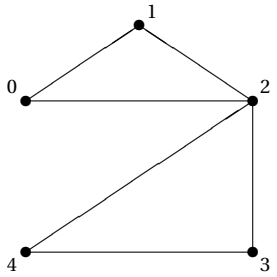
```c
#include <stdlib.h>
#include <stdio.h>

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 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);
```

```
42      p = p->next;
43    }
44    printf("Null\n");
45    return;
46  }
47
48  int main(void) {
49    struct nodo *primo;
50    primo = leggi_lista();
51    completa_lista(primo);
52    stampa_lista(primo);
53    return(0);
54  }
```

Esercizio n. 3

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'.

Codifica in linguaggio C

```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;
}

int leggi_grafo(struct nodo *v[]) {
  int i, n;
  printf("Numero di vertici del grafo: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("Lista di adiacenza del vertice %d\n", i);
    v[i] = leggi_lista();
  }
  return(n);
}

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

int appartiene(int i, struct nodo *t) {
  int trovato = 0;
  while (t!=NULL && !trovato) {
    if (t->info == i)
      trovato = 1;
    t = t->next;
  }
  return(trovato);
}

int copertura(struct nodo *v[], int n, struct nodo *p) {
  struct nodo *q;
  int i, flag=1;
  for (i=0; i<n && flag; i++) {
    flag = 0;
    if (!appartiene(i, p)) {
      q = v[i];
      flag = 1;
      while (flag && q!=NULL) {
        if (!appartiene(q->info, p))
          flag = 0;
        q = q->next;
      }
    } else {
      flag = 1;
    }
  }
  return(flag);
}

```

```
85 int main(void) {
86   struct nodo *v[MAX], *p;
87   int n;
88   n = leggi_grafo(v);
89   p = leggi_lista();
90   if (copertura(v, n, p))
91     printf("La lista e‘ una copertura del grafo.\n");
92   else
93     printf("La lista non e‘ una copertura del grafo.\n");
94   return(0);
95 }
```

Esercizio n. 4

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;
  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);
}


```

```
47 void stampa_grafo(struct nodo *L[], int n) {
48   int i;
49   printf("Liste di adiacenza dei vertici del grafo:\n");
50   for (i=0; i<n; i++) {
51     printf("%2d: ", i);
52     stampa_lista(L[i]);
53   }
54   printf("\n");
55   return;
56 }
57 
58 int adiacente(struct nodo *G[], int u, int v) {
59   struct nodo *p;
60   p = G[u];
61   while (p != NULL && p->info != v) {
62     p = p->next;
63   }
64   if (p == NULL)
65     return(0);
66   else
67     return(1);
68 }
69 
70 int main(void) {
71   struct nodo *G[100], *primo, *p;
72   int n, ok=1;
73   n = leggi_grafo(G);
74   stampa_grafo(G, n);
75   printf("Inserimento della lista di vertici da verificare\n");
76   primo = leggi_lista();
77   p = primo;
78   while (ok && p->next != NULL) {
79     if (!adiacente(G, p->info, p->next->info))
80       ok = 0;
81     p = p->next;
82   }
83   if (ok)
84     printf("La lista costituisce un cammino sul grafo.\n");
85   else
86     printf("La lista non rappresenta un cammino sul grafo.\n");
87   return(0);
88 }
```