

# Corso di Algoritmi e Strutture Dati (IN110)

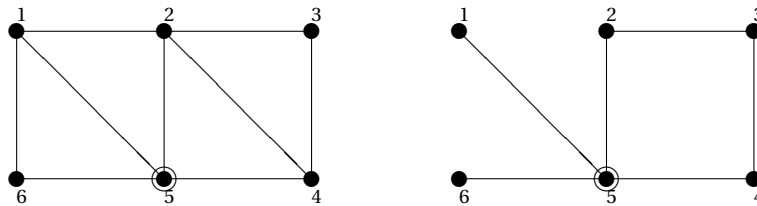
## Tutorato n. 12

Marco Liverani\*

### Esercizio n. 1

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

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #define MAX 30
4
5 struct nodo {
6     int info;
7     struct nodo *next;
8 };
9
10 struct nodo *leggi_lista(void) {
11     struct nodo *p, *primo;
12     int i, n;
13     printf(" inserisci il numero di elementi: ");
14     scanf("%d", &n);
```

\*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/>

```

15 printf(" inserisci %d elementi: ", n);
16 primo = NULL;
17 for (i=0; i<n; i++) {
18     p = malloc(sizeof(struct nodo));
19     p->next = primo;
20     scanf("%d", &p->info);
21     primo = p;
22 }
23 return(primo);
24 }
25
26 void stampa_lista(struct nodo *p) {
27     while (p != NULL) {
28         printf("%d --> ", p->info);
29         p = p->next;
30     }
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 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) {

```

```

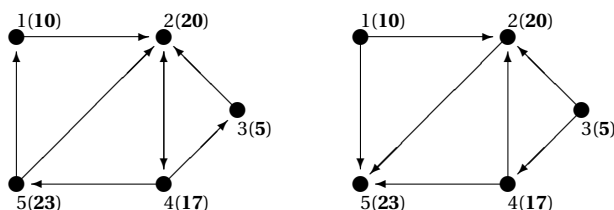
69     q = primo;
70     primo = primo->next;
71 } else {
72     p = primo;
73     while (p->next != NULL && p->next->info != v) {
74         p = p->next;
75     }
76     if (p->next != NULL) {
77         q = p->next;
78         p->next = p->next->next;
79     }
80 }
81 if (q != NULL)
82     free(q);
83 }
84 return(primo);
85 }
86
87 int main(void) {
88     struct nodo *G[MAX], *p, *q;
89     int n, gmax, vmax, v, g;
90     n = leggi_grafo(G);
91     gmax = grado(G[0]);
92     vmax = 0;
93     for (v=1; v<n; v++) {
94         g = grado(G[v]);
95         if (g > gmax) {
96             gmax = g;
97             vmax = v;
98         }
99     }
100     printf("Il vertice di grado massimo scelto e' %d.\n", vmax);
101     p = G[vmax];
102     while (p->next != NULL) {
103         q = p->next;
104         while (q != NULL) {
105             G[q->info] = elimina(p->info, G[q->info]);
106             G[p->info] = elimina(q->info, G[p->info]);
107             q = q->next;
108         }
109         p = p->next;
110     }
111     stampa_grafo(G, n);
112     return(0);
113 }

```

## Esercizio n. 2

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:  $\{w_1, \dots, w_n\}$ . 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

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #define MAX 50
4
5 struct nodo {
6     int info;
7     struct nodo *next;
8 };
9
10 struct nodo *leggi_lista(void) {
11     struct nodo *p, *primo;
12     int i, n;
13     printf(" inserisci il numero di elementi: ");
14     scanf("%d", &n);
15     printf(" inserisci %d elementi: ", n);
16     primo = NULL;
17     for (i=0; i<n; i++) {
18         p = malloc(sizeof(struct nodo));
19         p->next = primo;
20         scanf("%d", &p->info);
21         primo = p;
22     }
23     return(primo);
24 }
25
26 void stampa_lista(struct nodo *p) {
27     while (p != NULL) {
28         printf("%d --> ", p->info);
29         p = p->next;
30     }
```

```

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);

```

```

85  for (i=0; i<n; i++) {
86      p = G[i];
87      prec = NULL;
88      while (p != NULL) {
89          if (w[i] > w[p->info]) {
90              aggiungi(G, p->info, i);
91              if (prec != NULL) {
92                  prec->next = p->next;
93                  free(p);
94                  p = prec->next;
95              } else {
96                  G[i] = p->next;
97                  free(p);
98                  p = G[i];
99              }
100             } else {
101                 prec = p;
102                 p = p->next;
103             }
104         }
105     }
106     stampa_grafo(G, n);
107     return(0);
108 }

```

### Esercizio n. 3

Letto un grafo non orientato  $G = (V, E)$  e letta una lista di vertici di  $V$ ,  $L = \{v_1, \dots, v_k\}$ , 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.

#### Codifica in linguaggio C

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 struct nodo {
5     int info;
6     struct nodo *next;
7 };
8
9 struct nodo *leggiLista(void) {
10     struct nodo *p, *primo;
11     int i, n;
12     printf(" inserisci il numero di elementi: ");
13     scanf("%d", &n);
14     printf(" inserisci %d elementi: ", n);
15     primo = NULL;
16     for (i=0; i<n; i++) {
17         p = malloc(sizeof(struct nodo));
18         p->next = primo;
19         scanf("%d", &p->info);
20         primo = p;
21     }
22     return(primo);
23 }
24
25 void stampaLista(struct nodo *p) {
26     while (p != NULL) {
27         printf("%d --> ", p->info);
28         p = p->next;
29     }
30     printf("Null\n");
31     return;
32 }
33
34 int leggiGrafo(struct nodo *G[]) {
35     int i, n;
36     printf("Inserisci il 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         G[i] = leggiLista();
41     }
42     return(n);
43 }
44
45 void stampaGrafo(struct nodo *G[], int n) {
```

```

46  int i;
47  printf("Liste di adiacenza dei vertici del grafo:\n");
48  for (i=0; i<n; i++) {
49      printf(" vertici adiacenti a %d: ", i);
50      stampaLista(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[100], *L;
85     int n;
86     n = leggiGrafo(G);
87     stampaGrafo(G, n);
88     L = leggiLista();
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 }

```