Corso di Informatica 1 (IN1) – Prof. Marco Liverani – a.a. 2006/2007

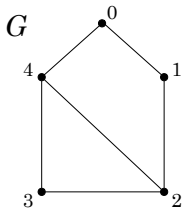
Seconda prova di esonero – 10 gennaio 2007

Esercizio n. 1

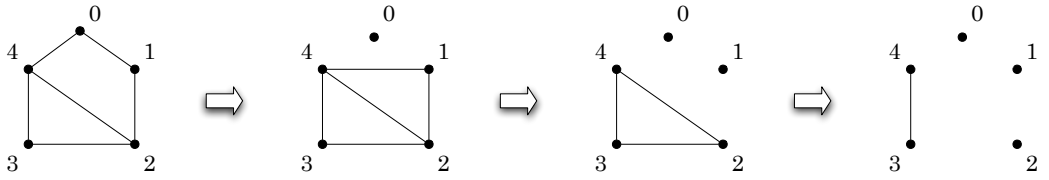
Risolvere il seguente problema proponendo una codifica completa del programma in linguaggio C.

Sia $G = (V, E)$ un grafo non orientato letto in input e rappresentato con liste di adiacenza. Eliminare, ripetutamente, gli spigoli incidenti sui vertici di grado 2 aggiungendo, se non esiste, uno spigolo tra i vertici adiacenti a quelli di grado 2; al termine dell'esecuzione del programma il grafo non deve contenere nessun vertice di grado 2. Stampare le liste di adiacenza del grafo "residuo".

Esempio Si consideri il grafo $G = (V, E)$ rappresentato in figura, con $V = \{0, 1, 2, 3, 4\}$ ed $E = \{(0, 1), (0, 4), (1, 2), (2, 3), (2, 4)\}$.



L'eliminazione dei vertici (e l'inserimento di nuovi spigoli) procede secondo i passaggi successivi rappresentati nella seguente figura (nota bene: la riduzione non è unica, dipende dall'ordine con cui vengono esaminati i vertici):



Soluzione

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

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

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

  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);
}

int leggi_grafo(struct nodo *G[]) {
  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);
    G[i] = leggi_lista();
  }
  return(n);
}

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 grado(int x, struct nodo *G[]) {
  int d=0;
  struct nodo *p;

  p = G[x];
  while (p != NULL) {
    d++;
    p = p->next;
  }
  return(d);
}
```

```
int adiacenti(struct nodo *G[], int x, int y) {
  struct nodo *p;
  int rc;

  p = G[x];
  while (p != NULL && p->info != y)
    p = p->next;
  if (p == NULL)
    rc = 0;
  else
    rc = 1;
  return(rc);
}

void aggiungi_spigolo(struct nodo *G[], int x, int y) {
  struct nodo *p;

  p = malloc(sizeof(struct nodo));
  p->info = y;
  p->next = G[x];
  G[x] = p;
  p = malloc(sizeof(struct nodo));
  p->info = x;
  p->next = G[y];
  G[y] = p;
  return;
}

void elimina_spigolo(struct nodo *G[], int x, int y) {
  struct nodo *p, *prec = NULL;

  p = G[x];
  while (p->info != y) {
    prec = p;
    p = p->next;
  }
  if (prec != NULL)
    prec->next = p->next;
  else
    G[x] = p->next;
  free(p);

  p = G[y];
  while (p->info != x) {
    prec = p;
    p = p->next;
  }
  if (prec != NULL)
    prec->next = p->next;
  else
    G[y] = p->next;
  free(p);

  return;
}
```

```
void riduzione(struct nodo *G[], int n) {
  int flag=0, i, u, v;

  while (flag == 0) {
    flag = 1;
    for (i=0; i<n; i++) {
      if (grado(i, G) == 2) {
        u = G[i]->info;
        v = G[i]->next->info;
        flag = 0;
        elimina_spigolo(G, i, u);
        elimina_spigolo(G, i, v);
        if (!adiacenti(G, u, v))
          aggiungi_spigolo(G, u, v);
      }
    }
  }
  return;
}

int main(void) {
  struct nodo *G[MAX];
  int n;

  n = leggi_grafo(G);
  printf("Liste di adiacenza del grafo originale:\n");
  stampa_grafo(G, n);
  riduzione(G, n);
  printf("Liste di adiacenza del grafo ridotto:\n");
  stampa_grafo(G, n);
  return(1);
}
```

Esercizio n. 2

Risolvere il seguente problema proponendo una codifica completa del programma in linguaggio C.

Letta in input una sequenza di numeri interi memorizzarla in una lista. Stampare la lista. Eliminare dalla lista tutti gli elementi "doppioni". Stampare la lista "residua".

Esempio Si consideri la seguente lista di elementi:

$$1 \to 2 \to 37 \to 2 \to 1 \to 1 \to 25 \to -5 \to 2$$

In seguito all'eliminazione dei "doppioni" la lista residua sarà la seguente:

$$1 \to 2 \to 37 \to 25 \to -5$$

Soluzione

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

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

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

  printf("  Numero di elementi: ");
  scanf("%d", &n);
  printf("  Inserisci %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");
  return;
}

void elimina_doppioni(struct nodo *primo) {
  struct nodo *p, *q, *prec;

  p = primo;
  while (p != NULL) {
    prec = p;
    q = p->next;
    while (q != NULL) {
      if (q->info == p->info) {
        prec->next = q->next;
        free(q);
        q = prec->next;
      } else {
        prec = q;
        q = q->next;
      }
    }
    p = p->next;
  }
  return;
}

int main(void) {
  struct nodo *p;

  p = leggi_lista();
  stampa_lista(p);
  elimina_doppioni(p);
  stampa_lista(p);
  return(0);
}
```