Corso di Informatica 1 (IN1) – Prof. Marco Liverani – a.a. 2007/2008

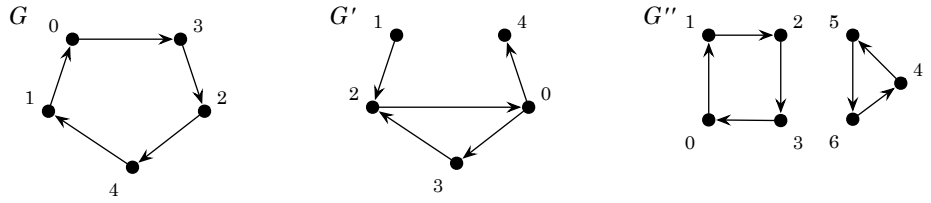
Seconda prova di esonero – 7 gennaio 2008

Risolvere i seguenti problemi proponendo, per ciascun esercizio, la codifica in linguaggio C di un programma completo.

Esercizio n. 1

Letto in input un grafo orientato $G = (V, E)$ con n vertici, memorizzarlo utilizzando delle liste di adiacenza. Verificare se il grafo è isomorfo ad un ciclo C_n orientato.

Esempio Si considerino i grafi rappresentati in figura. Dei tre grafi G, G' e G'', solo G è un ciclo, ossia un grafo isomorfo a C_5 orientato (nel caso del grafo G, infatti, risulta $n = |V| = 5$); gli altri due grafi, pur contenendo dei cicli, non sono isomorfi ad un C_n orientato.



Soluzione

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #define MAX 100
4
5  struct nodo {
6    int info;
7    struct nodo *next;
8  };
9
10 struct nodo *leggiLista(void) {
11   int i, n;
12   struct nodo *p, *primo=NULL;
13
14   printf("Numero di elementi: ");
15   scanf("%d", &n);
16   printf("Inserisci %d elementi: ", n);
17   for (i=0; i<n; i++) {
18     p = malloc(sizeof(struct nodo));
19     scanf("%d", &p->info);
20     p->next = primo;
21     primo = p;
```

[1] Per superare la prova di esonero è necessario ottenere almeno 15 punti; tuttavia affinché le prove di esonero siano valide è necessario che la media dei voti del primo e del secondo esonero sia maggiore o uguale a 18/30. È possibile utilizzare libri e appunti personali, senza scambiarli con altri studenti. I compiti che presenteranno evidenti ed anomale "similitudini" saranno annullati.

```
22    }
23    return(primo);
24  }
25
26  int leggiGrafo(struct nodo *G[]) {
27    int i, n;
28    printf("Numero di vertici del grafo: ");
29    scanf("%d", &n);
30    for (i=0; i<n; i++) {
31      printf("Lista di adiacenza del vertice %d:\n", i);
32      G[i] = leggiLista();
33    }
34    return(n);
35  }
36
37  int grado(struct nodo *p) {
38    int g = 0;
39    while (p != NULL) {
40      p = p->next;
41      g++;
42    }
43    return(g);
44  }
45
46  int verificaCiclo(struct nodo *G[], int n) {
47    int check[MAX], i, flag;
48    for (i=0; i<n; i++)
49      check[i] = 0;
50    i = 0;
51    while (check[i]==0 && grado(G[i])==1) {
52      check[i] = 1;
53      i = G[i]->info;
54    }
55    flag = 1;
56    if (i!=0) {
57      flag = 0;
58    } else {
59      for (i=0; i<n && flag==1; i++)
60        flag = check[i];
61    }
62    return(flag);
63  }
64
65  int main(void) {
66    struct nodo *G[MAX];
67    int n;
68    n = leggiGrafo(G);
69    if (verificaCiclo(G, n))
70      printf("Il grafo e' isomorfo ad un ciclo C%d.\n", n);
71    else
72      printf("Il grafo NON e' isomorfo ad un ciclo C%d.\n", n);
73    return(0);
74  }
```

Esercizio n. 2

Letta in input una sequenza di numeri interi memorizzarli in una lista in modo tale che, nella prima parte della lista, siano memorizzati tutti i numeri pari e, nella seconda parte della lista, siano memorizzati tutti i numeri dispari; in ciascuna delle due parti gli elementi devono essere memorizzati in ordine crescente. Stampare la lista.

Esempio Si consideri la seguente sequenza di interi letta in input: $3, 18, 2, 8, 4, 17, 6, 9, 5$; la lista che deve essere costruita è la seguente: $2 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 18 \rightarrow 3 \rightarrow 5 \rightarrow 9 \rightarrow 17 \rightarrow$ null.

Soluzione

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  struct nodo {
5    int info;
6    struct nodo *next;
7  };
8
9  void inserisci(struct nodo **primo, struct nodo **p) {
10   struct nodo *q;
11   if (*primo == NULL || (*primo)->info > (*p)->info) {
12     (*p)->next = *primo;
13     *primo = *p;
14   } else {
15     q = *primo;
16     while (q->next != NULL && q->next->info < (*p)->info)
17       q = q->next;
18     (*p)->next = q->next;
19     q->next = *p;
20   }
21   return;
22 }
23
24 struct nodo *leggiLista(void) {
25   int n, i;
26   struct nodo *p, *pari=NULL, *dispari=NULL;
27   printf("Numero di elementi: ");
28   scanf("%d", &n);
29   printf("Inserisci %d interi: ", n);
30   for (i=0; i<n; i++) {
31     p = malloc(sizeof(struct nodo));
32     scanf("%d", &p->info);
33     if (p->info % 2 == 0)
34       inserisci(&pari, &p);
35     else
36       inserisci(&dispari, &p);
37   }
38   if (pari == NULL) {
39     p = dispari;
40   } else {
41     p = pari;
42     while (p->next != NULL)
43       p = p->next;
44     p->next = dispari;
45     p = pari;
```

```
46      }
47      return(p);
48    }
49
50    void stampaLista(struct nodo *p) {
51      while (p != NULL) {
52        printf("%d --> ", p->info);
53        p = p->next;
54      }
55      printf("null\n");
56      return;
57    }
58
59    int main(void) {
60      struct nodo *primo;
61      primo = leggiLista();
62      stampaLista(primo);
63      return(0);
64    }
```