Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2014/2015

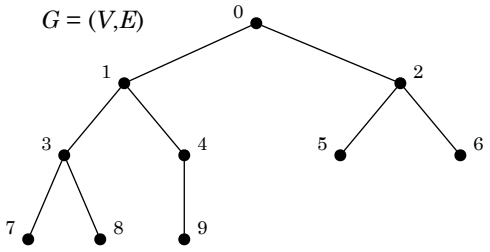
Seconda prova di esonero – 9 gennaio 2015

Risolvere i seguenti problemi proponendo, per ciascun esercizio, la codifica in linguaggio C di un programma completo. La prova dura tre ore, durante le quali non è possibile allontanarsi dall'aula, se non dopo aver consegnato l'elaborato scritto. 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.
Deve essere consegnata solo la "bella copia" del compito scritto; su ciascun foglio deve essere riportato il nome, il cognome e il numero di matricola (o un altro codice identificativo di fantasia) dello studente.

Esercizio n. 1

Letto in input un numero intero $n > 0$ costruire le liste di adiacenza di un albero binario completo $G = (V, E)$, con $V = \{0, 1, 2, \ldots, n-1\}$. Gli spigoli dell'albero sono non orientati. Stampare le liste di adiacenza di G.

Esempio Sia $n = 10$; il l'albero binario completo non orientato di cui devono essere costruite le liste di adiacenza è rappresentato nella figura seguente:



Soluzione

```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 100

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

void stampaLista(struct nodo *p) {
  while (p!=NULL) {
    printf("%d --> ", p->info);
    p = p->next;
  }
```

```
15    printf("NULL\n");
16    return;
17  }
18
19  void stampaGrafo(struct nodo *V[], int n) {
20    int i;
21    for (i=0; i<n; i++) {
22      printf("%2d: ", i);
23      stampaLista(V[i]);
24    }
25    return;
26  }
27
28  void aggiungiSpigolo(struct nodo *V[], int i, int j) {
29    struct nodo *p;
30    p = malloc(sizeof(struct nodo));
31    p->info = j;
32    p->next = V[i];
33    V[i] = p;
34    p = malloc(sizeof(struct nodo));
35    p->info = i;
36    p->next = V[j];
37    V[j] = p;
38    return;
39  }
40
41  void creaAlberoBinarioCompleto(struct nodo *V[], int n) {
42    int i;
43    for (i=0; i<n; i++) {
44      V[i] = NULL;
45    }
46    for (i=0; i<n; i++) {
47      if (2*i+1<n)
48        aggiungiSpigolo(V, i, 2*i+1);
49      if (2*i+2<n)
50        aggiungiSpigolo(V, i, 2*i+2);
51    }
52    return;
53  }
54
55  int main(void) {
56    int n;
57    struct nodo *V[MAX];
58    printf("Numero di vertici: ");
59    scanf("%d", &n);
60    creaAlberoBinarioCompleto(V, n);
61    stampaGrafo(V, n);
62    return(0);
63  }
```

Esercizio n. 2

Letta in input una sequenza di n numeri interi memorizzarla in una lista e calcolare la media aritmetica m tra l'elemento di valore massimo e l'elemento di valore minimo. Stampare l'elemento della lista con la minima differenza (in valore assoluto) dalla media m. Il valore assoluto di un numero floating point può essere calcolato con la funzione di libreria "`fabs(...)`".

Esempio Sia $n = 8$ e supponiamo che la lista, costituita da n numeri interi, sia la seguente:

$$27 \rightarrow 64 \rightarrow 3 \rightarrow 50 \rightarrow 82 \rightarrow 18 \rightarrow 53 \rightarrow 35$$

La media aritmetica tra l'elemento massimo e l'elemento minimo è data da $m = (3 + 82)/2 = 42,5$. L'elemento con lo scarto minimo da m è 50. Da notare che anche l'elemento 35 ha scarto dalla media minimo ($|m - 50| = |m - 35| = 7,5$): il programma si limita a stampare uno degli elementi con scarto minimo dalla media.

Soluzione

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

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

float media(struct nodo *p) {
  int min, max;
  float m=0;
  min = p->info;
  max = p->info;
  while (p != NULL) {
    if (min > p->info)
      min = p->info;
    if (max < p->info)
      max = p->info;
    p = p->next;
  }
  m = (float)(min+max)/2;
  return(m);
}

int minimoScarto(struct nodo *p, float m) {
  int min;
  min = p->info;
  while (p != NULL) {
    if (fabs(m - p->info) < fabs(m - min))
      min = p->info;
    p = p->next;
  }
  return(min);
}

```

```
37 struct nodo *leggiLista(void) {
38   struct nodo *p, *primo=NULL;
39   int i, n;
40   printf("Numero di elementi: ");
41   scanf("%d", &n);
42   printf("Inserisci %d numeri interi: ", n);
43   for (i=0; i<n; i++) {
44     p = malloc(sizeof(struct nodo));
45     scanf("%d", &p->info);
46     p->next = primo;
47     primo = p;
48   }
49   return(primo);
50 }
51
52 int main(void) {
53   int s;
54   float m;
55   struct nodo *p;
56   p = leggiLista();
57   m = media(p);
58   s = minimoScarto(p, m);
59   printf("L'elemento di scarto minimo dalla media m=%.2f e' %d\n", m, s);
60   return(0);
61 }
```