Algoritmi e Strutture Dati (IN110)
Esercitazione n. 8

Marco Liverani *

Esercizio n. 1

Leggere in input una sequenza di numeri interi terminata dal numero -1 e memorizzarla in una lista A. Costruire una nuova lista B "compressa" secondo lo standard RLE (*run length encoding*): per ogni elemento di A si memorizza in B l'elemento stesso e il numero di ripetizioni consecutive in A; i nodi della lista B saranno dunque costituiti almeno da due campi ciascuno. Stampare la lista B, terminata dal valore -1.

Ad esempio se $A = (3,3,3,3,2,2,3,5,5,5)$ allora $B = ((3,4),(2,2),(3,1),(5,3))$.

Codifica in linguaggio C

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

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

struct nodoRLE {
  int info, rip;
  struct nodoRLE *next;
};

struct nodo *leggiLista(void) {
  struct nodo *p, *primo = NULL;
  int x;
  scanf("%d", &x);
  while (x != -1) {
    p = malloc(sizeof(struct nodo));
    p->info = x;
    p->next = primo;
    primo = p;
    scanf("%d", &x);
  }
  return(primo);
}
```

*Università degli Studi Roma Tre, Corso di Laurea in Matematica, Corso di Algoritmi e Strutture Dati (IN110) – sito web del corso http://www.mat.uniroma3.it/users/liverani/IN110/

```
27
28 void stampaListaRLE(struct nodoRLE *p) {
29   while (p != NULL) {
30     printf("%d %d ", p->info, p->rip);
31     p = p->next;
32   }
33   printf("-1\n");
34   return;
35 }
36
37 struct nodoRLE *compressioneRLE(struct nodo *p) {
38   struct nodoRLE *q, *primo=NULL;
39   int prec = -1;
40   while (p != NULL) {
41     if (p->info == prec) {
42       q->rip++;
43     } else {
44       q = malloc(sizeof(struct nodoRLE));
45       q->info = p->info;
46       q->rip = 1;
47       q->next = primo;
48       primo = q;
49       prec = q->info;
50     }
51     p = p->next;
52   }
53   return(primo);
54 }
55
56 int main(void) {
57   struct nodo *p;
58   struct nodoRLE *q;
59   p = leggiLista();
60   q = compressioneRLE(p);
61   stampaListaRLE(q);
62   return(0);
63 }
```

Esercizio n. 2

Leggere in input una sequenza numeri interi "compressa" in formato RLE e terminata dal valore -1: la sequenza è costituita da un valore e dal numero di ripetizioni consecutive dello stesso valore. Memorizzare la sequenza in una lista A. Costruire e stampare la lista B corrispondente in formato "espanso". Esempio: se viene letta in input la sequenza $2,3,5,2,7,4,3,2,-1$ allora deve essere costruita la lista $B = (\underbrace{2,2,2}_{3 \text{ volte}}, \underbrace{5,5}_{2 \text{ volte}}, \underbrace{7,7,7,7}_{4 \text{ volte}}, \underbrace{3,3}_{2 \text{ volte}})$.

Codifica in linguaggio C

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

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

struct nodoRLE {
  int info, rip;
  struct nodoRLE *next;
};

struct nodoRLE *leggiListaRLE(void) {
  struct nodoRLE *p, *primo=NULL;
  int x;
  scanf("%d", &x);
  while (x != -1) {
    p = malloc(sizeof(struct nodoRLE));
    p->info = x;
    scanf("%d", &p->rip);
    p->next = primo;
    primo = p;
    scanf("%d", &x);
  }
  return(primo);
}

struct nodo *decompressioneRLE(struct nodoRLE *q) {
  int i;
  struct nodo *p, *primo=NULL;
  while (q != NULL) {
    for (i = 0; i < q->rip; i++) {
      p = malloc(sizeof(struct nodo));
      p->info = q->info;
      p->next = primo;
      primo = p;
    }
    q = q->next;
  }
  return(primo);
}
```

```
43
44 void stampaLista(struct nodo *p) {
45   while (p != NULL) {
46     printf("%d ", p->info);
47     p = p->next;
48   }
49   printf("-1\n");
50   return;
51 }
52
53 int main(void) {
54   struct nodo *p;
55   struct nodoRLE *q;
56   q = leggiListaRLE();
57   p = decompressioneRLE(q);
58   stampaLista(p);
59   return(0);
60 }
```

Esercizio n. 3

Letta in input una sequenza di numeri floating point la memorizza su una lista. Quindi stampa tutte le sotto-liste (prive di intersezione) in cui la somma degli elementi sia minore della media tra l'elemento massimo e minimo dell'intera lista.

Codifica in linguaggio C

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

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

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo=NULL;
  int i, n;
  float x;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d elementi: ", n);
  for (i=0; i<n; i++) {
    scanf("%f", &x);
    p = malloc(sizeof(struct nodo));
    p->info = x;
    p->next = primo;
    primo = p;
  }
  return(primo);
}

void stampa_lista(struct nodo *p) {
  while (p != NULL) {
    printf("%f --> ", p->info);
    p = p->next;
  }
  printf("NULL\n");
  return;
}

float max(struct nodo *p) {
  float m;
  m = p->info;
  while (p != NULL) {
    if (m < p->info)
      m = p->info;
    p = p->next;
  }
  return(m);
}


```

```
47 float min(struct nodo *p) {
48   float m;
49   m = p->info;
50   while (p != NULL) {
51     if (m > p->info)
52       m = p->info;
53     p = p->next;
54   }
55   return(m);
56 }
57 
58 void calcola(struct nodo *p) {
59   float media, somma;
60   media = (min(p) + max(p))/2.0;
61   somma = 0.0;
62   while (p != NULL) {
63     if (somma + p->info < media) {
64       somma = somma + p->info;
65       printf("%f ", p->info);
66       p=p->next;
67     } else {
68       printf("\n");
69       somma = 0.0;
70       if (p->info >= media)
71         p = p->next;
72     }
73   }
74   printf("\n");
75   return;
76 }
77 
78 int main(void) {
79   struct nodo *primo;
80   primo = leggi_lista();
81   stampa_lista(primo);
82   calcola(primo);
83   return(0);
84 }
```