Corso di Algoritmi e Strutture Dati (IN110) – Prof. Marco Liverani – a.a. 2022/2023

Esame scritto del 10 Febbraio 2023 (Appello B)

Si richiede di risolvere entrambi gli esercizi riportando una codifica in linguaggio C completa dei due programmi. Nel caso in cui non si riesca a completare entrambi gli esercizi si suggerisce di riportare almeno la codifica in C delle funzioni principali o una loro pseudo-codifica. È possibile consultare libri e appunti personali, ma non scambiare libri o appunti con altri studenti. I compiti che presenteranno evidenti ed anomale "similitudini" saranno annullati. La prova scritta ha una durata di tre ore, durante le quali non è consentito allontanarsi dall'aula, se non dopo aver consegnato il compito.
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 intero positivo n, costruire un array A con i primi n elementi della successione di Furbonacci, definita come segue:

$$\begin{aligned} a_0 &= 1 \\ a_i &= \begin{cases} a_0 + \cdots + a_{i-1} \text{ se } a_{i-1} \text{ è dispari} \\ \max\{a_0, \ldots, a_{i-1}\} - a_{i-2} \text{ altrimenti} \end{cases} \end{aligned}$$

Stampare in output l'array A.

Esempio Siano $n = 10$. L'array A con i primi 10 elementi della successione di Furbonacci è il seguente:

$$A = (1, 1, 2, 1, 5, 10, 5, 25, 50, 25)$$

Soluzione

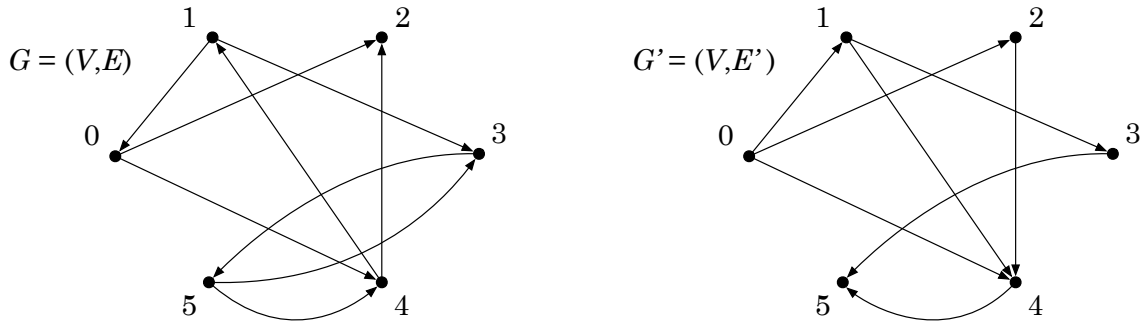
```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 50

int generaArray(int A[]) {
  int i, m, n, s;
  printf("Insersci n: ");
  scanf("%d", &n);
  A[0] = 1;
  s = 0;
  m = 1;
  for (i=1; i<n; i++) {
    s = s + A[i-1];
    if (m < A[i-1])
      m = A[i-1];
    if (A[i-1] % 2 == 1)
      A[i] = s;
    else
      A[i] = m-A[i-2];
  }
  return(n);
}
```

```
23
24 void stampaArray(int X[], int n) {
25   int i;
26   for (i=0; i<n; i++)
27     printf("%d ", X[i]);
28   printf("\n");
29   return;
30 }
31
32 int main(void) {
33   int A[MAX], n;
34   n = generaArray(A);
35   stampaArray(A, n);
36   return(0);
37 }
```

Esercizio n. 2

Leggere in input le liste di adiacenza di un grafo orientato $G = (V, E)$ con n vertici. Visualizzare in output le liste di adiacenza di G. Per ogni spigolo $(u, v) \in E(G)$, se $u > v$ eliminare lo spigolo da G e aggiungere, se non esiste già, lo spigolo (v, u). Visualizzare in output le liste di adiacenza del grafo modificato.

Esempio Si consideri il grafo G rappresentato in figura. Il grafo viene modificato dal programma ottenendo il grafo G' in figura.



Soluzione

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

```
14    }
15    printf("null\n");
16    return;
17  }
18
19  void stampaGrafo(struct nodo *V[], int n) {
20    for (int i=0; i<n; i++) {
21      printf("%2d: ", i);
22      stampaLista(V[i]);
23    }
24    return;
25  }
26
27  struct nodo *leggiLista(void) {
28    struct nodo *p, *primo = NULL;
29    int i, n;
30    printf("Numero di elementi: ");
31    scanf("%d", &n);
32    printf("inserisci %d elementi: ", n);
33    for (i=0; i<n; i++) {
34      p = malloc(sizeof(struct nodo));
35      scanf("%d", &p->info);
36      p->next = primo;
37      primo = p;
38    }
39    return(primo);
40  }
41
42  int leggiGrafo(struct nodo *V[]) {
43    int i, n;
44    printf("Numero di vertici: ");
45    scanf("%d", &n);
46    for (i=0; i<n; i++)
47      V[i] = leggiLista();
48    return(n);
49  }
50
51  int adiacenti(struct nodo *V[], int i, int j) {
52    int r;
53    struct nodo *p;
54    p = V[i];
55    while (p != NULL && p->info != j)
56      p = p->next;
57    if (p == NULL)
58      r = 0;
59    else
60      r = 1;
61    return(r);
62  }
63
64  void aggiungiSpigolo(struct nodo *V[], int i, int j) {
65    struct nodo *p;
66    if (!adiacenti(V, i, j)) {
67      p = malloc(sizeof(struct nodo));
68      p->info = j;
69      p->next = V[i];
```

```
70      V[i] = p;
71    }
72    return;
73  }
74
75  void eliminaSpigolo(struct nodo *V[], int i, int j) {
76    struct nodo *p, *q = NULL;
77    p = V[i];
78    while (p != NULL && p->info != j) {
79      q = p;
80      p = p->next;
81    }
82    if (q == NULL)
83      V[i] = p->next;
84    else
85      q->next = p->next;
86    free(p);
87    return;
88  }
89
90  void modificaGrafo(struct nodo *V[], int n) {
91    struct nodo *p;
92    int i, j;
93    for (i=0; i<n; i++) {
94      p = V[i];
95      while (p != NULL) {
96        if (p->info < i) {
97          j = p->info;
98          p = p->next;
99          aggiungiSpigolo(V, j, i);
100         eliminaSpigolo(V, i, j);
101       } else
102         p = p->next;
103     }
104   }
105   return;
106 }
107
108 int main(void) {
109   struct nodo *G[MAX];
110   int n;
111   n = leggiGrafo(G);
112   stampaGrafo(G, n);
113   modificaGrafo(G, n);
114   stampaGrafo(G, n);
115   return(0);
116 }
```