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

Esame scritto del 22 gennaio 2007

Esercizio n. 1

Letto in input un numero intero $n > 0$ generare in modo casuale un array A di n interi compresi tra $-n$ e n (estremi inclusi). Stampare il vettore A. Inserire nel vettore A gli elementi mancanti per completare la sequenza (crescente o decrescente) di numeri interi tra a_i e a_{i+1}. Stampare il vettore A completato.

Esempio Sia $n = 8$ e sia $A = (-3, 2, 8, 7, 5, 8, 4, -2)$ la sequenza di numeri casuali compresi tra -8 e 8. L'array A viene completato nel modo seguente (in grassetto gli elementi "originali", per distinguerli dagli elementi inseriti per completare il vettore):

$$A = (-\mathbf{3}, -2, -1, 0, 1, \mathbf{2}, 3, 4, 5, 6, 7, \mathbf{8}, \mathbf{7}, 6, \mathbf{5}, 6, 7, \mathbf{8}, 7, 6, 5, \mathbf{4}, 3, 2, 1, 0, -1, -\mathbf{2})$$

Soluzione

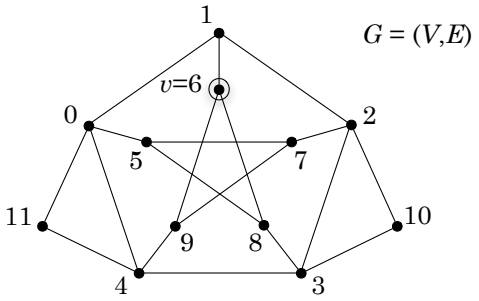
```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 100

int genera_array(int V[]) {
  int i, n;

  printf("Inserisci un numero intero positivo: ");
  scanf("%d", &n);
  srand((unsigned)time(NULL));
  for (i=0; i<n; i++)
    V[i] = rand() % (2*n+1) - n;
  return(n);
}

void stampa_array(int V[], int n) {
  int i;

  for (i=0; i<n; i++)
    printf("%d ", V[i]);
  printf("\n");
  return;
}

void inserisci(int X[], int n, int i, int y) {
  int j;

  for (j=n; j>i+1; j--)
    X[j] = X[j-1];
  X[j] = y;
  return;
}
```

```
int completa(int V[], int n) {
  int i;

  for (i=0; i<n-1; i++) {
    if (V[i]-V[i+1] > 1) {
      inserisci(V, n, i, V[i]-1);
      n++;
    } else if (V[i]-V[i+1] < -1) {
      inserisci(V, n, i, V[i]+1);
      n++;
    }
  }
  return(n);
}

int main(void) {
  int A[MAX], n;

  n = genera_array(A);
  stampa_array(A, n);
  n = completa(A, n);
  stampa_array(A, n);
  return(0);
}
```

Esercizio n. 2

Letto un grafo non orientato $G = (V, E)$ con n vertici, rappresentarlo con liste di adiacenza. Letta inoltre in input un vertice $v \in V(G)$ costruire la lista L dei vertici $u \in V(G)$ a distanza 2 da v, in altri termini si chiede di costruire la lista di tutti i vertici u non adiacenti a v, ma adiacenti ad un vertice adiacente a v. Stampare la lista L.

Esempio Si consideri il grafo $G = (V, E)$ rappresentato in figura.



Sia $v = 6$ il vertice letto in input. La lista L dei vertici a distanza 2 da v è la seguente:

$$L : 0 \to 2 \to 3 \to 4 \to 5 \to 7 \to \text{NULL}$$

Soluzione

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

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

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

  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d elementi: ", 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 dei vertici adiacenti al 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 adiacenti(struct nodo *G[], int v, int u) {
  struct nodo *p;
  int rc;
  p = G[v];
  while (p != NULL && p->info != u)
    p = p->next;
  if (p != NULL)
    rc = 1;
  else
    rc = 0;
  return(rc);
}

int esiste(struct nodo *p, int x) {
  int rc;
  while (p != NULL && p->info != x)
    p = p->next;
  if (p == NULL)
    rc = 0;
  else
    rc = 1;
  return(rc);
}

struct nodo *costruisci_lista(struct nodo *G[], int v) {
  struct nodo *p, *q, *r, *primo=NULL;

  p = G[v];
  while (p != NULL) {
    q = G[p->info];
    while (q != NULL) {
      if (q->info != v && !adiacenti(G, v, q->info) && !esiste(primo, q->info)) {
        r = malloc(sizeof(struct nodo));
        r->info = q->info;
        r->next = primo;
        primo = r;
      }
      q = q->next;
    }
    p = p->next;
  }
  return(primo);
}

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

  n = leggi_grafo(G);
  printf("Inserisci un vertice del grafo: ");
  scanf("%d", &v);
  L = costruisci_lista(G, v);
  printf("Liste di adiacenza del grafo G:\n");
  stampa_grafo(G, n);
  printf("Lista dei vertici a distanza 2 da %d:\n", v);
  stampa_lista(L);
  return(0);
}
```