Corso di Informatica 1 (IN1 - Fondamenti) – Prof. Marco Liverani – a.a. 2005/2006

# Esame scritto del 22 settembre 2006

## Esercizio n. 1

Letto in input un numero intero $n > 0$ generare in modo casuale un array $A$ di $n$ interi compresi tra $-5n$ e $3n$. Stampare il vettore $A$. Stampare il numero di elementi della più lunga sottosequenza non decrescente contenuta nel vettore $A$.

**Esempio** Sia $n = 9$ e sia $A = (-3, 17, -32, -10, 5, 12, 4, 22, 23)$ la sequenza di numeri casuali compresi tra $-45$ e 27. È facile riconoscere che in questo caso la più lunga sottosequenza di numeri non decrescenti è composta da 4 elementi: $-32, -10, 5, 12$.

### Soluzione

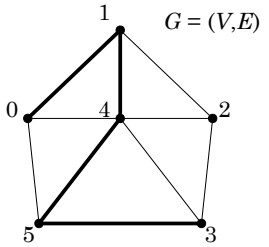
```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 500

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

  srand((unsigned)time(NULL));
  for (i=0; i<n; i++) {
    V[i] = rand() % (8*n + 1) - 5*n;
  }
  return;
}

int max_lunghezza(int n, int V[]) {
  int i, max_temp, max;

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

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

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

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

  printf("Numero di elementi: ");
  scanf("%d", &n);
  array_casuale(n, A);
  stampa_array(n, A);
  m = max_lunghezza(n, A);
  printf("La sottosequenza non decrescente di lunghezza massima ");
  printf("e' costituita da %d elementi.\n", m);
  return(0);
}
```

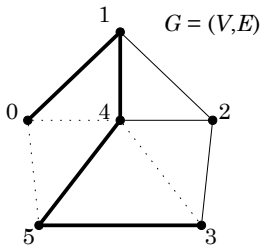
# Esercizio n. 2

Letto un grafo non orientato $G = (V, E)$ con $n$ vertici, rappresentarlo con liste di adiacenza. Letta inoltre una sequenza di vertici di $V(G)$ che rappresenta un cammino $\mathcal{P}$ su $G$, memorizzarla in una lista. Effettuare la riduzione transitiva di $\mathcal{P}$ su $G$, eliminando dal grafo tutti gli spigoli $(u, v) \in E(G)$ tali che $u, v \in V(\mathcal{P})$ e $(u, v) \notin E(\mathcal{P})$. Stampare le liste di adiacenza del grafo "ridotto".

**Esempio** Si consideri il grafo $G = (V, E)$ rappresentato in figura ed il cammino $\mathcal{P} = (0, 1, 4, 5, 3)$ evidenziato su di esso.




Devono essere eliminati da $G$ tutti gli spigoli che non appartengono al cammino e che uniscono i vertici del cammino. In questo caso quindi saranno eliminati dal grafo $G$ gli spigoli $(0, 4)$, $(0, 5)$, $(3, 4)$, ottenendo il grafo rappresentato nella figura seguente.



## Soluzione

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

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

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

void stampaGrafo(struct nodo *G[], int n) {
  int i;

  for (i=0; i<n; i++) {
    printf("  %d: ", i);
    stampaLista(G[i]);
  }
  return;
}

struct nodo *leggiLista(void) {
  int i, n;
  struct nodo *p, *primo;

  primo = NULL;
  printf("  Numero di elementi: ");
  scanf("%d", &n);
  printf("  Inserisci gli elementi della lista: ");
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->next = primo;
    primo = p;
    scanf("%d", &p->info);
  }
  return(primo);
}

int leggiGrafo(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] = leggiLista();
  }
  return(n);
}
```

```
void eliminaSpigolo(int u, int v, struct nodo *G[]) {
  struct nodo *p, *q;

  p = G[u];
  q = NULL;
  while (p != NULL && p->info != v) {
    q = p;
    p = p->next;
  }
  if (p != NULL) {
    if (q == NULL)
      G[u] = p->next;
    else
      q->next = p->next;
    free(p);
  }
  return;
}

void riduzioneTransitiva(struct nodo *L, struct nodo *G[], int n) {
  struct nodo *p, *q;

  p = L;
  while (p->next != NULL && p->next->next != NULL) {
    q = p->next->next;
    while (q != NULL) {
      eliminaSpigolo(p->info, q->info, G);
      eliminaSpigolo(q->info, p->info, G);
      q = q->next;
    }
    p = p->next;
  }
  return;
}

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

  n = leggiGrafo(G);
  printf("\nListe di adiacenza del grafo:\n");
  stampaGrafo(G, n);
  printf("\nLista dei vertici del cammino:\n");
  L = leggiLista();
  riduzioneTransitiva(L, G, n);
  printf("\nListe di adiacenza del grafo in seguito alla riduzione:\n");
  stampaGrafo(G, n);
  return(0);
}
```