Appunti di informatica

Lezione 10

anno accademico 2016-2017

Mario Verdicchio

Esercizio

- Scrivere un programma che, data una sequenza di 10 interi (scelta dall'utente), la ordini in ordine crescente

Soluzione 2: Bubble sort

- L'algoritmo "bubble sort" fa emergere come bolle i numeri più grandi e funziona come segue:
  1. si confrontano i primi 2 numeri e se il secondo è più piccolo del primo, si scambiano;
  2. si ripete il passo 1 per il secondo e terzo numero, per il terzo e quarto, e così via fino alla fine
  3. si ripetono i passi 1-2 per tante volte quanti sono i numeri nella sequenza (per essere sicuri che la sequenza sia ordinata)

Codice Bubble Sort

```
from __future__ import print_function
v = []
for i in range(0,10):
    v.append(input("inserisci un numero: "))
print ("ecco i numeri che hai inserito: ", v)
for j in range(0,10):
    for i in range(0,9):
        if v[i] > v[i+1]:
            z = v[i]
            v[i] = v[i+1]
            v[i+1] = z
print ("ecco i numeri ordinati:\n", v)
```

Possibile miglioramento

- Sia nel Selection sort, sia nel Bubble sort le operazioni di ordinamento vengono ripetute un numero fisso di volte, che dipende dalla dimensione della sequenza di numeri
- Nel Bubble sort in certi casi tali ripetizioni sono inutili (immaginate di ricevere una sequenza già in ordine crescente)
- Il Bubble sort può essere modificato inserendo una variabile che indica quando durante un ciclo di confronti non viene effettuato nessuno scambio (il che significa che la sequenza è ordinata); le ripetizioni nel Bubble sort devono continuare finché tale condizione non si verifica; in questo modo alla peggio facciamo tante ripetizioni quante nella versione attuale dell'algoritmo, ma nei casi più fortunati possiamo terminare prima.
- Una tale modifica non è possibile nel Selection sort perché in un suo ciclo non confrontiamo tra di loro i vari numeri della sequenza, ma un solo numero con tutti gli altri: se anche non ci sono scambi non possiamo dire che la sequenza sia già in ordine (ma solo che il numero selezionato è il più piccolo).

Codice Bubble Sort modificato

```
from __future__ import print_function
v = []
for i in range(0,10):
    v.append(input("inserisci un numero: "))
print ("ecco i numeri che hai inserito: ", v)
for j in range(0,10):
    scambio = 0
    for i in range(0,9):
        if v[i] > v[i+1]:
            z = v[i]
            v[i] = v[i+1]
            v[i+1] = z
            scambio = 1
    if scambio == 0:
        break
print ("ecco i numeri ordinati: ", v)
```

Variabili booleane

- Nel codice precedente la variabile “scambio” assume valore 0 per indicare che non c’è stato uno scambio (viene azzerata a ogni iterazione) e assume il valore 1 appena c’è stato uno scambio.
- Alla fine di un’iterazione, se “scambio” vale 1 vuol dire che bisogna farne un’altra, mentre se vale 0 vuol dire che la lista è stata già ordinata
- Al posto di valori numerici, “scambio” può essere usata con valori di verità (True e False)
- Variabili che assumono questi ultimi valori si dicono “booleane” in onore del logico inglese George Boole, che nel XIX ha dato un fortissimo impulso allo studio della logica (la disciplina degli operatori come AND, OR, etc.) in rapporto all’algebra

Codice Bubble Sort modificato con variabile booleana

```
from __future__ import print_function
v = []
for i in range(0,10):
    v.append(input("inserisci un numero: "))
print ("ecco i numeri che hai inserito: ", v)
for j in range(0,10):
    scambio = False
    for i in range(0,9):
        if v[i] > v[i+1]:
            z = v[i]
            v[i] = v[i+1]
            v[i+1] = z
            scambio = True
    if not scambio:
        break
print ("ecco i numeri ordinati: ", v)
```

Esercizio

- Scrivere un programma in Python che, dato un numero in base 10 in input, dia in output la sua codifica binaria in complemento a due.

Per creare la sequenza di bit

```
while True:
    v.insert(0, x % 2)
    if x/2 == 0:
        break
    else:
        x = x/2
#i resti della divisione vengono inseriti in
#prima posizione, quindi l'ordine dei bit è
#già inverso quando l'array si completa
#si esce dal while quando la divisione dà 0.
```

Soluzione completa (1)

```
from __future__ import print_function
n = input("insersici un numero da convertire in binario\n")
v = []

if n < 0:
    x = -n
else:
    x = n

while True:
    v.insert(0, x % 2)
    if x/2 == 0:
        break
    else:
        x = x/2
v.insert(0,0)
#inseriamo uno 0 all'inizio per indicare il segno positivo
```

Soluzione completa (2)

```
#se il numero originario è negativo dobbiamo invertire tutti
#i bit dopo il primo “1” a partire da destra
if n < 0:
    j = len(v)-1
#il seguente ciclo while individua la posizione del primo “1” da destra
    while True:
        if v[j] == 1:
            break
        else:
            j = j-1
#il seguente ciclo for inverte tutti i bit che sono a sinistra del primo “1” da destra
    for i in range(0,j):
        if v[i] == 0:
            v[i] = 1
        else:
            v[i] = 0
print(v)

for i in range(0,len(v)):
    print(v[i], end ="")
print ("\n")
```