Calcolo rapido dei Numeri di Fibonacci con il linguaggio Python

Giuseppe Matarazzo

Maggio 2012

Sommario

In altre occasioni, in passato, è capitato di utilizzare degli artifici per il calcolo dei numeri di Fibonacci con migliaia di cifre. Mai con la potenza esecutiva e l'intrinseca eleganza del metodo presentato in questa memoria, che utilizza gli stringatissimi comandi del linguaggio per computer *Python*.

Introduzione

La serie dei **Numeri di Fibonacci**, che gode di una ben nota *proprietà armonica*, si ottiene sommando i due numeri interi precedenti quello esaminato. La procedura di calcolo è di tipo *ricorsivo* a partire dai primi due elementi F(0)=0 e F(1)=1; da cui si ottiene F(2)=1, F(3)=2 e così via.

La routine base di calcolo è estremamente semplice. Eccola:

```
def myfib(n):
... a,b = 0, 1
... for i in range (n):
... ... a,b = b, a+b
... return a
```

Verrà esposto un metodo di raggruppamento delle migliaia di cifre del numero di Fibonacci F(n) in modo che esse siano leggibili su video e su file. I *tripli puntini* stanno a indicare l'indentazione del codice Python, che nel listato non appaiono.

Prima opzione: Calcolo di F(n)

Si ricava solo il numero finale della serie di Fibonacci F(n); il tempo di esecuzione di Python è dell'ordine di grandezza di qualche millisecondo.

Come si vede nel prossimo riquadro, l'algoritmo può essere scritto come una funzione di n, che viene richiamata dalla linea di comando Python (>>>). Si importa il file della funzione e si calcola il valore della serie F(n) con il comando indicato. Al risultato, comprendente tutte le cifre, viene accodata la lettera L per indicare che è stata superata la riga classica delle 80 colonne.

```
# ------ fb.py ------
def myfib(n):
   a,b = 0, 1
   for i in range (n):
      a,b = b, a+b
   return a

'''
C:\>python
Python 2.6.5 (r265:79096, Mar 19 2010, 21:48:26) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import fb
>>> fb.myfib(3000)
41061588630797126033356837871926710522012510863736925240888543092690558427411340
37313304916608500445608300368357069422745885693621454765026743730454468521604866
06292497360503469773453733196887405847255290082049086907512622059054542195889758
03110922267084927479385953913331837124479554314761107327624006673793408519173181
09932017067768389347667647787395021744702686278209185538422258583064083016618629
00358266857238210235802504351951472997919676524004784236376453347268364152648346
24584057321424141993791724291860263981009786694239201540462015381867142573983507
48513964211399827136406795811784581986586922859680432436567097960000L
>>> fb.myfib(0)
0
>>> fb.myfib(1)
1
>>> fb.myfib(2)
1
>>> fb.myfib(3)
2
>>> quit()
'''
```

Si esce dall'ambiente Python digitando *quit()*.

Vediamo adesso come fare per raggruppare un numero F(n) di Fibonacci a migliaia di cifre, dividendolo per esempio in 8 colonne per riga. Non è questo il luogo per spiegare come fa Python ad effettuare una *manovra* simile, ci sono in rete innumerevoli *tutorial* a cui rimandiamo. Ci limitiamo solo a esporre algoritmo e output

```
# ------------- BestFibo.py --------------
# Copia F(n), con n molto elevato su video, su file e incolonnato
# a gruppi di 8 colonne per riga
# ----------------------------------------
import sys
sys.setrecursionlimit(5000)
'''
 Consente di ampliare l'intervallo
 di ricorsivita' da 999 al limite della
 memoria del PC (variabile); scelto il valore 5000
```

```
'''
def myfib(n):
   a,b = 0, 1
   for i in range (n):
      a,b = b, a+b
   return a

import itertools

def split_columns(iterable, size):
    c = itertools.count()
    for k, g in itertools.groupby(iterable, lambda item: c.next() // size):
        yield list(g)

def pretty(number, size=8, per_line=8):
    groups = split_columns(split_columns(str(number), size), per_line)
    lines = (' '.join(''.join(group) for group in line) for line in groups)
    return '\n'.join(lines)

numero = input("  Numero Fibonacci F(n) = ")
s=myfib(numero)
print pretty(s)
# --- sotto: scrittura risultato su file --------
outfile = open('Fibonacci-' + str(numero) +'.txt', 'w+')
outfile.write("F(" + str(numero) + str(") =") + '\n')
outfile.write(pretty(s))
outfile.close()
# ------- fine BestFibo.py --------------

'''
File creato dal programma: Fibonacci-5000.txt

F(5000) =
38789684 54388325 63370191 63083259 05312082 12771464 62451061 60597214
89555013 90440370 97010822 91646221 06694792 93452858 88297381 34831020
08954982 94036143 01569114 78938364 21656394 41069102 14505634 13370655
86562382 54656700 71252592 99038549 33813928 83637834 75189087 62970712
03333705 29231076 93008518 09384980 18038478 13996748 88176555 46537882
91644268 91298038 46137789 69021502 29308247 56663462 24923071 88332480
32803750 39130352 90330450 58427011 47635242 27021093 46376991 04006714
17488329 84228914 91273104 05432875 32980442 73676822 97724498 77498745
55691907 70388063 70468327 94811358 97373999 31101062 19308149 01857081
53978543 79195305 61751076 10530756 88783766 03366735 54452588 44886241
61921055 34574936 75897849 02798823 43510235 99844663 93485325 64119522
21859563 06047536 46454707 60330902 42080638 25849291 56452876 29157575
91423438 09142302 91749108 89841552 09854432 48659407 97935713 16841692
86803954 53095453 88698114 66508206 68628974 20639323 43848846 52409887
42395873 80197699 38203171 74208932 26546887 93640026 30797780 05875912
96713896 34214252 57911687 27556003 60311370 54775472 46046399 87588046
98517840 86743828 63125
```

```
,,,
```

Seconda opzione: Calcolo di tutta la serie fino a F(n)

In questo caso il codice Python assume la forma di lista di dati, e, ad ogni ciclo, il programma stampa l'intera serie di Fibonacci fino al raggiungimento del numero n desiderato:

```
# ------------ fiblist.py ---------------------------------------
# Risultato diretto della lista:
# import fiblist oppure da riga di comando: python fiblist.py
#
# Per la scrittura su file: python fiblist.py > nomefile.txt
# ricordarsi di dare l'input, es.30, perché il prompt non appare)
# ---------------------------------------------------------------
def fibonacci():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

def firstn(g, n):
for i in range(n):
yield i, g.next()

def print_list(list):       # scrive una riga per ciclo [i, F(i)]
   for e in list: print e


numero = input("  Numero finale della serie Fibonacci F(n) = " + '\n')
s=list(firstn(fibonacci(), numero))
print print_list(s)


,,,
Comando per risultati su file: python fiblist.py > 30.txt
  Numero finale della serie Fibonacci F(n) = 30
(0, 0)
(1, 1)
(2, 1)
(3, 2)
(4, 3)
(5, 5)
(6, 8)
(7, 13)
(8, 21)
(9, 34)
(10, 55)
(11, 89)
(12, 144)
(13, 233)
```

```
(14, 377)
(15, 610)
(16, 987)
(17, 1597)
(18, 2584)
(19, 4181)
(20, 6765)
(21, 10946)
(22, 17711)
(23, 28657)
(24, 46368)
(25, 75025)
(26, 121393)
(27, 196418)
(28, 317811)
(29, 514229)
None = fine ciclo
'''
```

Conclusione

E' significativo osservare come l'algoritmo esposto sia superiore alle strutture di altri linguaggi di programmazione (C++, Fortran) non solo per la sua *intrinseca eleganza* ma anche per la velocità nel raccogliere e conservare i risultati ottenuti.

..... 23-05-2012 .....