OliCyber.IT 2023 - Approfondimenti
Introduzione a SageMath

Contenuti

1 Introduzione a SageMath

SageMath[1] è un sistema di algebra computazionale (Computer Algebra System in inglese, CAS[2]) open source. In poche parole è in grado di manipolare espressioni matematiche in modo molto simile a quello che un essere umano farebbe su carta (ossia evitare il più possibile di "svolgere i calcoli"): ad esempio, $\frac{2}{6}$ viene rappresentato internamente come uno specifico oggetto *numero razionale* semplificato ai minimi termini ($\frac{1}{3}$) anziché come `float(1/3) == 0.3333333333333333`. Questo, oltre talvolta a evitare errori di approssimazione, si trasforma nella possibilità di gestire oggetti matematici astratti piuttosto complessi utilizzando una sintassi ad alto livello che consente di manipolarli senza dover scendere in molti dettagli implementativi.
Il software è immenso: oltre a fornire implementazioni di vari algoritmi, funge anche da interfaccia alle principali librerie matematiche open source allo stato dell'arte come GMP[3], NTL[4], PARI[5] e molte altre.
Nonostante la maggior parte delle sue funzionalità siano avanzate (è uno dei coltellini svizzeri utilizzati da ricercatori e universitari per testare congetture e risultati), ci sono alcune feature di base che possono essere estremamente utili anche a un soggetto inesperto.
Ovviamente utilizzare algoritmi alla cieca non è mai didatticamente utile; ciononostante, accettare che esista codice efficiente che risolve un dato problema può permetterti di ragionare più ad alto livello sulle tematiche che dovrai affrontare. Se la curiosità dovesse schiacciarti, il fatto che sia open source implica che tutti i sorgenti sono a tua disposizione scavando a sufficienza nella documentazione o nei file di installazione! Provare ad analizzarli è sicuramente molto impegnativo, ma senza dubbio altrettanto costruttivo.
Let's dive in!

1.1 Installazione

SageMath è piuttosto pesante; puoi testare brevi programmi su SageCell[6], un'interfaccia web che permette di utilizzare il software (con alcune limitazioni legate a problemi di security: il codice che scrivi dev'essere autocontenuto - senza quindi accesso ad internet).
Ti consiglio vivamente di utilizzare SageCell per cominciare a giocarci.
Nel caso in cui dovesse piacerti e volessi installarlo sulla tua macchina, qui[7] c'è la guida dedicata: la maggior parte delle distribuzioni Linux lo rende disponibile direttamente dal package manager.
Su Ubuntu, ad esempio (avendo i permessi di root):

```
$ sudo apt install sagemath sagemath-jupyter sagemath-doc-en
```

Dopo l'installazione, ci sono quattro modi per utilizzare il software:

- console interattiva, simile a quella di Python, digitando da terminale il comando `sage` (assumendo che la cartella in cui vengono installati i binari sia in `PATH`)
- lanciare un notebook con il comando `sage -n jupyter`
- scrivere uno script a sè stante **mio_script.sage**, come un programma in Python, ricordandoti di *utilizzare la sintassi di Sage*, da lanciare con il comando da terminale `sage mio_script.sage` (consigliato)
- scrivere uno script a sè stante **in Python**, importando l'intera libreria

```
from sage.all import *
# poi faccio cose
```

Oltre alla ricerca dei sorgenti, dalla console interattiva è possibile accedere al sorgente di una particolare classe / di una particolare funzione molto facilmente, digitando due punti interrogativi dopo la parola chiave di cui vuoi sapere di più:

[1] https://doc.sagemath.org/html/en/reference/
[2] https://en.wikipedia.org/wiki/Computer_algebra_system
[3] https://gmplib.org/
[4] https://libntl.org/
[5] https://en.wikipedia.org/wiki/PARI/GP
[6] https://sagecell.sagemath.org/
[7] https://7896a56df78170d5bab0f306d1a7230986a4206a–sagemath-tobias.netlify.app/installation/index.html

```
$ sage
 ----------------------------------------------------
|SageMath version 9.x, Release Date: xxxx-xx-xx    |
|Using Python 3.x.xx. Type "help()" for help.      |
|____________________________________________________|
sage: Zmod??
```

Vedrai comparire anche il percorso assoluto del file sorgente, così da poter andare a recuperarlo con precisione.

1.2 Sintassi e funzioni di base

Come avrai potuto intuire, Sage è costruito sopra Python. Nonostante ci siano alcune differenze sottili (e molto fastidiose in fase di apprendimento), puoi praticamente dare per scontato di star scrivendo in Python, debuggando eventuali errori facendo un uso considerevole di ricerche nella documentazione / su Google e della funzione `type()` per controllare con che oggetti stai avendo a che fare.
Questo perché Sage fa (spesso convenientemente) conversioni implicite degli oggetti che vengono utilizzati nelle operazioni (come Python, d'altronde), ma talvolta possono dare fastidio.
Per esempio, come ormai saprai bene in Python `a = 4; b = 2.0; c = a/b` farà sì che la variabile **c** risulti di tipo `float` nonostante il numeratore sia un `int`.
Sage ragiona allo stesso modo, con oggetti più complicati. Un esempio tra quelli che incontrerai più spesso sono gli interi in modulo. Prova ad eseguire il seguente codice su SageCell:

```
#Sage
n = 35; Zn = Zmod(n); m = Zn(3); m_inv = m^(-1)
print(3^(-1));
print(type(m)); print(m_inv, m*m_inv);
print(35*m); print(186*m)
```

Come avrai notato, il simbolo "^" in Sage rappresenta l'esponenziazione e non lo xor (Python). Ma c'è di più: le operazioni su `m` sono state contestualizzate al suo tipo, ossia un intero modulo 35. Questo vuol dire che volendo in Sage, se si va a definire il contesto opportuno per una certa variabile, ci si può dimenticare di inserirlo a mano ad ogni operazione! L'equivalente in Python sarebbe stato:

```
#Python
n = 35; m = 3; m_inv = pow(m, -1, n)
print(m*m_inv %n) #dobbiamo aggiungere noi il modulo!
print(35*m %n); print(186*m %n)
```

Questo era un esempio piuttosto banalotto, ma appena il codice comincia a diventare più complesso avere a disposizione una sintassi più pulita aumenta di molto la leggibilità; di conseguenza effettuare il debug diventa molto più semplice.
Passiamo ora a qualche funzione standard. Ricordi il Teorema Cinese del Resto (CRT) e l'algoritmo di Euclide Esteso (XGCD)? In Python è stato necessario implementarli da zero. Ebbene, in Sage questi sono algoritmi di default!

```
#Sage
a, b = 18, 12
g, x, y = xgcd(a,b)
print(f'{g = }, {x*a + y*b = }')

resti = [3, 8]; moduli = [11, 13]
x = crt(resti, moduli)
print(f'{x = }, {x%11 = }, {x%13 = }')
```

Vuoi cercare la fattorizzazione di un numero intero, ma non hai indizi riguardo un algoritmo ad hoc che abbia più possibilità degli altri di funzionare? No problem! Sage è equipaggiato anche per questo:

```
#Sage
n = randint(0,10^60) #grandino, no?
factor(n)
```

Aspetta, quanto veloce è?

```
#Sage
timeit("n=randint(0,10^60);factor(n)")
```

Whoa, ma quindi questa magia risolve il problema della fattorizzazione intera? Purtroppo no, gli algoritmi sono solo implementati molto bene (in particolare `factor()` chiama la backend di PARI, scritta in C, con GMP per la gestione dei bigint)...
Se volessi provare a far eseguire un certo comando con un timeout, si può fare uso degli `alarm` (presenti anche in Python):

```
from cysignals.alarm import alarm, AlarmInterrupt, cancel_alarm
n = randint(0,10^60); timeout = 0.200
try:
    alarm(timeout)
    print(factor(n))
except AlarmInterrupt:
    print("Timed out")
else:
    cancel_alarm()
```

Un'ultima cosa che può essere particolarmente sfiziosa è l'implementazione dei polinomi:

```
#Sage
n = 109
#definiamo l'anello dei polinomi nella variabile x a
#coefficienti negli interi modulo n
R.<x> = PolynomialRing(Zmod(n))
#si possono usare anche più variabili, per esempio R.<x,y>
pol = x + 1
print(pol^n) #si arrangia a fare tutto!
#possiamo dirgli di cambiare l'anello dei coefficienti
pol = pol.change_ring(ZZ)
print(pol^n)
#infine, in alcuni casi possiamo dirgli di cercare le radici..
R.<x> = PolynomialRing(RR)
pol = x^2 - x - 1
print(pol.roots())
```

In sostanza, ogni qualvolta ti ritrovi innanzi a un problema di matematica, se Python non riesce a rispondere alle tue necessità da solo, prova a dare una chance a Sage. Anche solo una ricerca del tipo "sage" + "nome problema" può essere illuminante.