Teoría de números

- Software y lenguajes para cálculo en Teoría de Números

Rudik Rompich

24 de noviembre de 2023

Universidad del Valle de Guatemala

Introducción

¿Por qué usar este tipo de métodos computacionales? ¿Cuál es la necesidad de usar software y lenguajes de cálculo en teoría de números?

- Algunos problemas en teoría de números involucran cálculos demasiado complejos para cálculos manuales.
- Estas herramientas permiten a los matemáticos explorar nuevos territorios y conjeturas al probar y visualizar rápidamente una amplia gama de casos.
- Plataformas de código abierto como SAGE Math facilitan la colaboración entre matemáticos de todo el mundo.
- Asistentes de prueba como Coq y Lean aseguran que cada paso lógico sea verificado, reduciendo significativamente el riesgo de errores.

Un momento...

¿Por qué no usamos simplemente **Python**?

¿Por qué no usamos simplemente Python?

- Python es un lenguaje de programación de propósito general y no está diseñado para matemáticas. (Las funciones implementadas nativamente no son las más eficientes)
- Sin embargo, muchos desarrolladores han creado bibliotecas de Python para matemáticas, como **NumPy** y **SciPy**.
- Y también programas como PARI/GP y SAGE Math tiene integración con Python.

Índice

Contexto Histórico/Herramientas

Comparaciones de velocidad

- Python vs PARI/GP
- Interpretación

Asistente de pruebas

- Lean

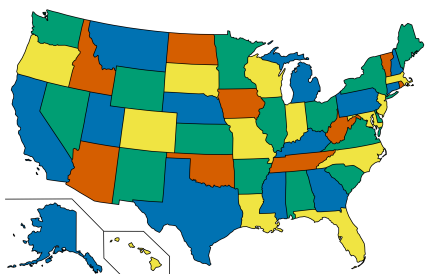
Contexto Histórico/Herramientas

Contexto Histórico/Herramientas

- **1950s**: Se desarrolla Fortran, considerado el primer lenguaje de programación de alto nivel.
- **1958**: Se desarrolla LISP, un lenguaje de programación basado en notación matemática para computadoras.
  - Se basa en el cálculo lambda, un sistema formal de matemática. Por ejemplo si f es una abstracción lambda y a es un término lambda, entonces fa es una aplicación de f sobre a ($f(a)$). Por ejemplo: $(\lambda x.x + 2)$

Contexto Histórico/Herramientas

- **1980s**
  - **Coq**, un asistente de prueba para teoría matemáticas y pruebas formales, fue introducido por el INRIA.
    - Prueba de los 4 colores



  - PARI/GP en la universidad de Bordeaux, enfocándose en la computación de teoría de números.

Contexto Histórico/Herramientas - PARI/GP

PARI/GP

- Se enfoca en la teoría de números algebraica y curvas elípticas.
- Se compone de la libreria PARI (C) y GP como lenguaje de scripting
  - Eficiencia en computación
  - Aritmética de polinomicos, operaciones de matrices, etc.
  - Open-source y desarrollo colaborativo

PARI/GP

- Problemas famosos
  - Conjectura de Birch y Swinnerton-Dyer

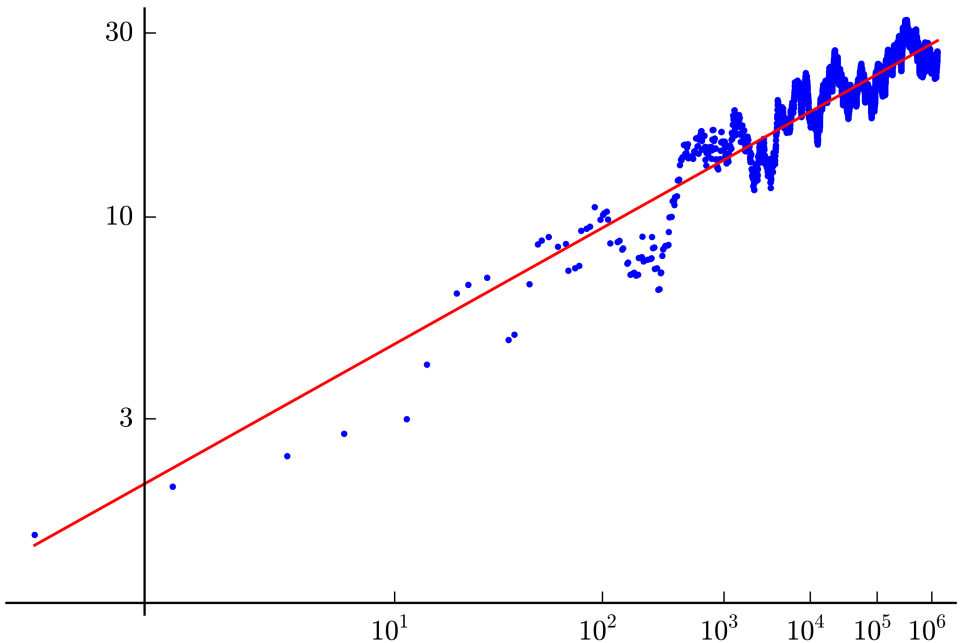


Figura 1: Conjunto de soluciones racionales de una ecuación de una curva elíptica. $y^2 = x^3 - 5x$

Contexto Histórico/Herramientas - PARI/GP

Ejemplos

- Generando números primos

```
forprime(p = 2, 100, print(p))
```

- Aritmética modular

```
Mod(7^50, 13)
```

- Congruencias lineales $3x \equiv 4$ mód 7

```
solve(Mod(3, 7)*x == Mod(4, 7), x)
```

Contexto Histórico/Herramientas - PARI/GP

Ejemplos

- Tripletas pitágoricas. (a, b, c). En donde $a^2 + b^2 = c^2$, donde $c = 25$.

```
 for(a=1, 24, for(b=a, 24,
if(a^2 + b^2 == 25^2, print([a, b, 25]))))
```

Contexto Histórico/Herramientas - PARI/GP

Problema interesante

Problema
Encontrar el primo más pequeño después de 10^{1000}.

Contexto Histórico/Herramientas - PARI/GP

Problema interesante

```
nextprime(10^20)
```

Contexto Histórico/Herramientas

- **1990s**
  - **Magma**, un software para álgebra, teoría de números, geometría algebraica, desarrollado por la University of Sydney.
  - $ 2800 por licencia

Contexto Histórico/Herramientas - SageMath

- **2004**
  - **SageMath**, un software diseñado por William Stein. Se basa en utilizar python (o cython) como interfaz y hace llamados a otras librerías; sin necesidad de aprender un nuevo lenguaje.
  - Surge como una alternativa a Magma, Maple, Mathematica, y MATLAB.

Contexto Histórico/Herramientas - SageMath

Mathematics packages contained in SageMath[17]	Algebra	GAP, Singular, FLINT
	Algebraic geometry	Singular
	Arbitrary-precision arithmetic	GMP, MPFR, MPFI, NTL, mpmath, Arb
	Arithmetic geometry	PARI/GP, NTL, mwrank, ECM
	Calculus	Maxima, SymPy, GiNaC, Giac, FriCAS
	Combinatorics	Symmetrica, Sage-Combinat
	Linear algebra	ATLAS, BLAS, LAPACK, NumPy, LinBox, IML, GSL
	Graph theory	NetworkX
	Group theory	GAP
	Numerical computation	GSL, SciPy, NumPy, ATLAS
	Number theory	PARI/GP, FLINT, NTL
	Statistical computing	R, SciPy
Other packages contained in SageMath	Command-line shell	IPython
	Database	ZODB, SQLite
	Graphical interface	SageMath Notebook, MathJax[18] (formerly jsMath)
	Graphics	matplotlib, Tachyon, GD, Jmol
	Interactive programming language	Python
	Networking	Twisted
Other Mathematics package available for SageMath	Differential geometry and tensor calculus	Sage Manifolds

Contexto Histórico/Herramientas

- **2013**
  - **Lean**, Lean fue desarrollado por Leonardo de Moura en el Microsoft Research.
  - El lenguaje está basado en Çalculus of Inductive Constructions"(CIC). Usa *higher-order logic* con un sistema poderoso de scripting.

Contexto Histórico/Herramientas - Lean

- **Problemas famosos usando Lean**
  - **Espacios perfectoides**: Uno de los logros importantes de Lean fue la formalización de una parte del trabajo de Peter Scholze sobre espacios perfectos, un concepto de geometría algebraica. Este proyecto demostró la capacidad de Lean para manejar tecnologías avanzadas y abstractas.
  - **Experimento del tensor líquido**: Este es un proyecto en curso destinado a formalizar una prueba del Experimento del Tensor Líquido propuesto por Scholze. Es un caso de prueba de la viabilidad de formalizar investigaciones de vanguardia en matemáticas utilizando asistentes de prueba como Lean.
  - **Problema Cap Set**: Lean se utilizó para formalizar una solución al problema del conjunto de límites, un problema de combinatoria. La prueba formal siguió de cerca el trabajo innovador de Jordan Ellenberg y Dion Gijswijt sobre el uso del método polinomial en combinatoria.

Contexto Histórico/Herramientas - Lean

Suma de 2 números

```
def add (x y : N) : N := x + y

#eval add 3 5  -- Outputs 8
```

Contexto Histórico/Herramientas - Lean

Adición es conmutativo

```
theorem add_comm (a b : N) : a + b = b + a :=
begin
  apply nat.rec_on b,
    rw [add_zero, zero_add],
  intro n,
  intro ih,
  rw [add_succ, succ_add, ih],
end
```

Contexto Histórico/Herramientas - Lean

Problema: Tenemos 2 enteros pares $a, b \implies a + b$ también es par

```
import data.int.basic
-- Definition of even integers
def is_even (n : Z) := \exists k, n = 2 * k
-- Theorem: Sum of two even numbers is even
theorem sum_of_even_is_even (a b : Z) (ha : is_even a)
 (hb : is_even b) : is_even (a + b) :=
begin
  -- Extract witnesses ‘k‘ and ‘l‘ such that ‘a = 2*k‘
  and ‘b = 2*l‘
  cases ha with k hk,
  cases hb with l hl,
  -- Express ‘a + b‘ using ‘k‘ and ‘l‘
  use (k + l),
  rw [hk, hl], - replace ‘a‘ with ‘2*k‘ and ‘b‘ with ‘2*l‘ ring,
end
```

Comparaciones de velocidad

Comparaciones de velocidad

- Para este apartado la idea es comparar Python vs PARI/GP usando SageMath.
- Demostración en código de las funciones

Generador de los primeros n primos

N	Python Time (s)	PARI/GP Time (s)
10	0.000209808349609375	1.3912551403045654
100	4.291534423828125e-06	0.0031762123107910156
1000	2.574920654296875e-05	0.000576019287109375
10000	8.392333984375e-05	0.001386880874633789
100000	0.0007810592651367188	0.0010728836059570312
1000000	0.0032541751861572266	0.0034050941467285156
10000000	0.03106379508972168	0.03308582305908203
100000000	0.22541308403015137	0.32268190383911133
1000000000	46.845951080322266	3.7162177562713623

Factorización de números

Python Time (s)	PARI/GP Time (s)
0.056957960128784 18	0.04412794113159 18
0.022554159164428 71	0.018842203063964 84
0.182974100112915 04	0.179669857025146 48
0.033399105072021 484	0.001436948776245 1172
105.63011693954468	109.23352813720703
61.27135896682739	70.22545719146729
38.07634902000427	42.30085301399231

Primeros números de Fibonacci

N (Fibonacci Index)	Python Time (s)	PARI/GP Time (s)
1000	0.0003190040588378906	0.002766847610473633
10000	2.4080276489257812e-05	0.000560760498046875
100000	0.0003409385681152344	0.0005068778991699219
1000000	0.012056827545166016	0.002707958221435547
10000000	0.04448366165161133	0.03272581100463867
100000000	0.6470000743865967	0.6016421318054199
1000000000	5.941126108169556	6.00231671333313

Función totiente

Value	Python Time (s)	PARI/GP Time (s)
1000	0.0003628730773925781	0.007293224334716797
10000	1.5974044799804688e-05	0.00180816650390625
100000	6.9141387939453125e-06	0.0007550716400146484
1000000	6.9141387939453125e-06	0.0008680820465087891
10000000	5.9604644775390625e-06	0.0007948875427246094
100000000	6.9141387939453125e-06	0.0010421276092529297
1000000000	5.9604644775390625e-06	0.0010669231414794922

Asistente de pruebas

Demostración de teorema de teoría de números usando Lean

Referencias

- Batut, C., Belabas, K., Bernardi, D., Cohen, H., & Olivier, M. (2000). User's Guide to PARI-GP. Université de Bordeaux I.
- https://adam.math.hhu.de
- https://lean-lang.org/theorem_proving_in_lean4/
- https://www.sagemath.org
- https://wiki.sagemath.org/DocumentationProject
- https://sci-hub.se/10.1038/d41586-021-01627-2