Números primos

INE5429 - Segurança em Computação

Caique Rodrigues Marques
c.r.marques@grad.ufsc.br

- O teste de primalidade de Fermat define que se p é primo e a não é divisível por p, então temos:

$$a^{p-1} \equiv 1 \pmod{p}. \tag{1}$$

  Para testar se p é primo, então vários a's aleatórios e não divisíveis por p são coletados. Se a igualdade em 1 não for válida com os valores definidos de a e p, então é dito que p é composto. Por outro lado, se a igualdade for válida para um ou mais valores de a, então p é um possível primo. No entanto, o teste aprova determinados pseudoprimos[1], principalmente os números de Carmichael[2].

- O teste de primalidade de Miller-Rabin é um aperfeiçoamento ao teste de primalidade de Fermat. Dado um n primo e $n > 2$, então $n - 1$ vai ser par e pode ser escrito como $2^s d$, onde s e d são positivos inteiros e d é ímpar. Podemos verificar que n não é primo, se para todo $0 \leq r \leq s - 1$ temos

$$a^d \not\equiv 1 \pmod{n} \wedge a^{2^r d} \not\equiv -1 \pmod{n},$$

  então n não é primo, caso contrário, n possivelmente é primo. Vale notar que a é uma testemunha de que n é composto, assim, a's são escolhidos aleatoriamente dentre o intervalo $1 < a < n - 1$.

- Para o caso do teste de Miller-Rabin, apesar de possuir uma complexidade maior e, consequentemente, possuir um maior tempo de execução, ele possui melhor precisão em detectar possíveis primos porque independe do valor de entrada, conseguindo uma probabilidade de $\frac{1}{2}$ de detectar um número composto. Ao contrário do caso do teste de Fermat, ele consegue atingir a mesma probabilidade se o valor de entrada não for um número de Carmichael, portanto, ele acaba tendo um caso claro de falso-positivo.

- O arquivo `primality_test.py` implementa ambos os testes de primalidade. Ambos podem ser usados da seguinte maneira:

```
$ python3
>>> from primality_test import fermat, miller_rabin
>>> miller_rabin(643808006803554439230129854961492699151386l0, 100)
True
>>> fermat(12764787846358441471, 100)
True
```

  O algoritmo de Miller-Rabin usa exponenciação modular, possuindo uma complexidade de $O(k \log^3 n)$, onde k é o número de testes realizados. O algoritmo de Fermat, que também usa exponenciação modular, possui uma complexidade de $O(k \times \log^2 n \times \log(\log(n)) \times \log(\log(\log(n))))$, onde k é o número de testes realizados.

- O arquivo `find_primes.py` é um simples programa que executa testes de primalidade, procurando por números primos com bits de tamanho 100, 200, 300, ..., 10000. A saída gerada apresenta o tempo para achar o número primo, o número de bits e o inteiro primo encontrado. Como é possível notar, o programa demora mais para achar primos com bits maiores que 2000, devido às várias exponenciações e operações modulares que são executadas.

[1] Os números p que satisfazem a condição em 1, mas não são primos, são chamados de pseudoprimos de Fermat.
[2] Número de Carmichael é um número composto p que satisfaz a relação em 1 para todos os inteiros a que são coprimos a p.