Relatório TC2 - Teste de Primalidade

Ian Resende - Vitor Basso

Triângulo de Pascal

Para realizar um teste de primalidade baseado no triângulo de pascal, foi utilizada a seguinte fórmula:

$$\binom{n}{p} = C_{n,p} = \frac{n!}{p!(n-p)!}$$

A fórmula calcula um coeficiente binomial que é um número que pertence a n-ésima linha do triângulo na posição p.

No código feito, utilizei uma função que calcula mais rapidamente esse coeficiente binomial. Para fazer o teste de primalidade de um número n, deve-se testar a divisibilidade de todos os coeficientes da linha n pelo número n, caso todos sejam divisíveis por n então n é primo.

Pode-se simplificar esse algoritmo comparando apenas da posição 3 da linha n até a posição n/2.

O teste feito dessa maneira se mostrou extremamente lento conforme o número a ser testado aumentava, por isso um limite de comparações foi imposto, o algoritmo agora verifica a divisibilidade do coeficiente binomial por n apenas da posição 3 até a 1000, apesar de não dar 100% de certeza no resultado ele continua satisfatório e agora conseguindo analisar números de centenas de dígitos em tempo razoável.

Pequeno teorema de fermat

O pequeno teorema de fermat oferece um meio de testar a primalidade de um número por meio da seguinte fórmula:

Pequeno Teorema de Fermat

Seja p um primo e $a \in \mathbb{Z}$. Então

$$a^p \equiv a \pmod{p}.$$

Em particular, se $p \nmid a$, então

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

Um algoritmo simples pode ser criado para testar a primalidade de um número se baseando nesse teorema. Um exemplo seria:

```
bool ehPrimo(p):
x= randomInt()
temp = x**p - x
retorna (temp % p)==0
```

Utilizando a linguagem python, tentei implementar uma função de exponenciação mais eficiente que a padrão, porém mesmo utilizando o método de exponenciação modular ou o método de calcular o x^2, x^4, x^8… e em seguida multiplicar esses resultados até chegar em x^p, a função padrão do python se mostrou mais eficiente.

Uma medida tomada para permitir analisar os maiores números possíveis foi deixar o x como sendo sempre 2 permitindo maiores números na potência.

Para conseguir ainda mais eficiência o número a ser testado é dividido em " i " partes e se eleva x(2) ao resultado inteiro dessa divisão e aplica o módulo de p, em seguida o resultado é elevado a " i ", multiplicado por x elevado ao resto da divisão anterior e o módulo de p novamente aplicado, essa medida tornou a análise de números grandes possível e muito mais rápida.

```
def power(p):
    i = 100000
    div = p//i
    res = (2**int(div))%p
    res = (res**i)*(2**(p%i))%p
    return res - 2
```

Análise de Tempo e tamanho dos números

Nos testes de comparação de tempo, primeiramente foi utilizado uma lista com números primos de 3 dígitos até 100 dígitos. O teste por fermat conseguiu analisar números até 16 dígitos enquanto o teste pelo triângulo de pascal conseguiu analisar rapidamente números de até 100 dígitos, devido a limitação imposta de analisar apenas os 1000 primeiros termos da linha do triângulo de pascal.

No teste pelo teorema de fermat, ocorre uma divisão do número a ser testado por um termo para permitir alcançar maiores valores de p. Assim esse termo foi alterado conforme o tamanho do número para otimizar os cálculos, com números de 3 a 12 dígitos o melhor termo para dividir é 10^4, entre 13 e 15 digitos o termo 10^5 se mostrou eficiente, e para 16 dígitos o termo 10^6 possibilitou a análise.

No universo dos números testados, o teste pelo triângulo de pascal analisando os primeiro 1000 termos da linha, foi consistente e rápido na análise.

Teste pelo Triângulo de Pascal – Primos

Dig	Número Primo	Seconds
3	101	0.000
4	1013	0.068
5	10007	0.532
6	100019	0.833
7	1000033	0.960
8	10000079	1.114
9	100030001	1.249
10	1500450271	1.790
11	10000000019	1.910
12	100123456789	2.112
13	1000000000039	2.303
14	10000000000283	2.481
15	100033000330001	3.207
16	1000000000100011	3.412
20	12764787846358441471	5.064
30	Primo de 30 Dígitos	9.813
40	Primo de 40 Dígitos	16.326
50	Primo de 50 Dígitos	23.970
80	Primo de 80 Dígitos	64.021
100	Primo de 100 Dígitos	94.703

Teste teorema de fermat - Números Primos

Divisão	Dig	Número Primo	Seconds
10^4	3	101	0.000
	4	1013	0.000
	5	10007	0.000
	6	100019	0.002
	7	1000033	0.007
	8	10000079	0.020
	9	100030001	0.024
	10	1500450271	0.031
	11	10000000019	0.051
	12	100123456789	0.260
	13	1000000000039	2.470
	14	10000000000283	26.990
10^5	13	1000000000039	1.334
	14	10000000000283	3.820
	15	100033000330001	28.840
10^6	16	1000000000100011	105.892

Analisando o resultado das duas funções é possível perceber que para números de até 13 dígitos o algoritmo do teorema de fermat faz uma análise em bem menos tempo, porém a partir dos 13 dígitos ele perde sua eficiência quase que exponencialmente, enquanto o triângulo de pascal se mantém respondendo em poucos segundos até 20 dígitos e consegue responder em um bom tempo até na casa das centenas de dígitos.

Teste teorema de fermat - Compostos

Divisão	Dig	Número Composto	Seconds
10^4	3	105	0.0
	4	1015	0.0
	5	10005	0.0
	6	100015	0.001
	7	1000055	0.011
	8	10000075	0.015
	9	100030005	0.019
	10	1500450275	0.028
	11	10000000015	0.042
	12	100123456785	0.234
	13	1000000000095	2.434
10^5	14	10000000000285	3.749
	15	100033000330005	30.474
10^6	16	1000000000100015	98.290

Teste pelo Triângulo de Pascal – Compostos

Dig	Número Composto	Seconds
3	105	0.000
5	10005	0.000
10	1500450275	0.000
15	100033000330005	0.000
30	12764787846358441473	0.000
40	Primo de 40 Dígitos	0.000
50	Primo de 50 Dígitos	0.000
80	Primo de 80 Dígitos	0.000
100	Primo de 100 Dígitos	0.000

Quando se analisa números compostos ímpares o algoritmo de fermat apresenta resultados parecidos com os números primos, enquanto o algoritmo de pascal é extremamente rápido mesmo com um número de centenas de dígitos, isso ocorre porque o algoritmo para assim que o primeiro termo testado não divide o n, ou seja, antes de atingir o máximo de termos analisados(1000).

Análise de erro

Para verificar a ocorrência de erros nos dois testes de primalidade, foram analisados todos os números de 3 a 10000, onde deveriam haver 1229 primos, e os resultados foram:

Algoritmo Pequeno Teorema de Fermat

No algoritmo de fermat, sem utilizar um parâmetro de confiança e mantendo o x como sempre o número 2, foram encontrados 1251 primos entre 2 e 10000, porém analisando somente os números primos nesse mesmo algoritmo todos passaram no teste, o que indica que achamos 31 falso-primos entre 2 e 10000.

Adicionando um parâmetro de confiança ao mesmo algoritmo, os resultados foram diferentes. Dessa vez o algoritmo analisava cada número 11 vezes com um número diferente como x, e o número só era considerado primo caso mais de 60% dos resultados apontasse como primo. Dessa maneira ao analisar o universo de inteiros de 2 até 10000 obtivemos 1235 primos, e ao analisar o universo de primos entre 2 e 10000 obtivemos 1226 primos, ou seja, o algoritmo deixou de reconhecer 3 primos, e obteve 9 falso-primos entre 2 e 10000.

Algoritmo Triângulo de Pascal

No método do triângulo de Pascal, quando se compara a divisibilidade de todos os termos da n-ésima linha com o n, sempre terá um resultado correto, porém no algoritmo estamos analisando apenas os primeiros 1000 termos o que abre uma brecha para possíveis falso-primos. Para analisar a ocorrência de erros, o limite de termos analisados foi diminuído para 300, pois até o número 300 o resultado será sempre correto, então foi feita uma análise no universo de inteiros de 300 até 10000, onde se obteve 1169 primos, porém existem apenas 1167 nesse intervalo, indicando que foram obtidos 2 falsos primos. A taxa de erro nessa análise foi pequena pois os números analisados eram relativamente pequenos, é evidente que números primos não serão dados como compostos nesse algoritmo, porém o contrário pode acontecer e possivelmente acontece com uma taxa maior em números de muitos dígitos.