Universidade Federal de Campina Grande

Ciência da Computação

Algoritmo mais eficiente para o cálculo de π

Disciplina

Laboratório de Organização e Arquitetura de Computadores

Professor

Elmar Melcher

elmar@dsc.ufcg.edu.br

Nome

Kleber Sobrinho

Matrícula: 119210988.

kleber.sobrinho@ccc.ufcg.edu.br

Campina Grande – PB

Julho de 2021

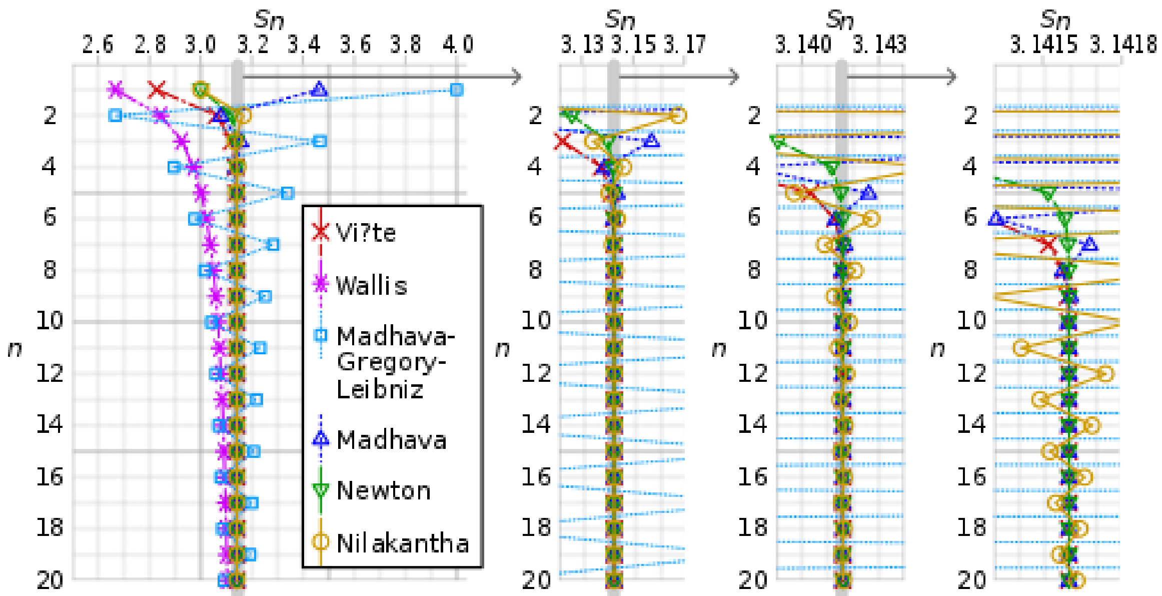
Premissa - Use uma série que converge mais rapidamente do que a série de Leibniz para calcular π e deixe rodar por uma hora, um fim de semana, ou mais. Quantos dígitos se consegue?

Série de Leibniz

A série de Leibniz para cálculo de π recebeu esse nome em homenagem a Gottfried Leibniz, matemático, filósofo, cientista e diplomata alemão. A série afirma que:

$$1 - 1/3 + 1/5 - 1/7 + 1/9 - \dots = \pi/4$$

A série de Leibniz converge de forma extremamente lenta para o cálculo de π, como pode ser visto na imagem abaixo, que compara a convergência da série de Leibniz com várias outras séries históricas infinitas para cálculo de π.



fonte: https://en.wikipedia.org/wiki/File:Comparison_pi_infinite_series.svg

O cálculo de π para 10 casas decimais corretas usando a soma direta da série requer cerca de cinco bilhões de termos.

Algoritmo de Gauss-Legendre

Apresenta uma convergência rápida, com apenas 25 interações produz 45 milhões de dígitos corretos de π. A desvantagem é que ele consome muita memória do computador.

O método é baseado no trabalho individual de Carl Friedrich Gauss (1777-1855) e Adrien-Marie Legendre (1752-1833) combinado com algoritmos modernos para multiplicação e raízes quadradas.

O algoritmo possui convergência quadrática, o que essencialmente significa que o número de dígitos corretos dobra a cada iteração do algoritmo.

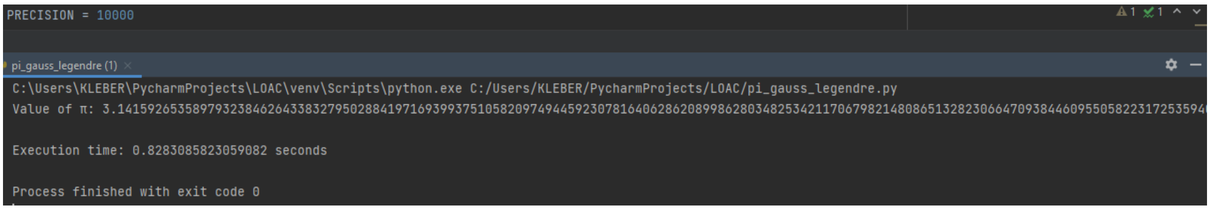
Implementação para o algoritmo

```
import decimal
import time

PRECISION = 10000


def calculate_pi_gauss_legendre():
    D = decimal.Decimal
    with decimal.localcontext() as ctx:
        ctx.prec += 2

        a, b, t, p = 1, 1 / D(2).sqrt(), 1 / D(4), 1  #initial value setting

        pi = None

        while True:
            an = (a + b) / 2
            b = (a * b).sqrt()
            t -= p * (a - an) * (a - an)
            a, p = an, 2*p

            pi_old = pi
            pi = (a + b) * (a + b) / (4 * t)

            if pi == pi_old:
                break

    return +pi

start_time = time.time()

decimal.getcontext().prec = PRECISION
print(f'Value of π: {calculate_pi_gauss_legendre()}')

end_time = time.time()
time_process = end_time - start_time

print(f'Execution time: {time_process}')
```

Modificando o valor da variável *PRECISION,* na quarta linha, do código acima podemos alterar a quantidade de casas decimais de π que serão calculadas.

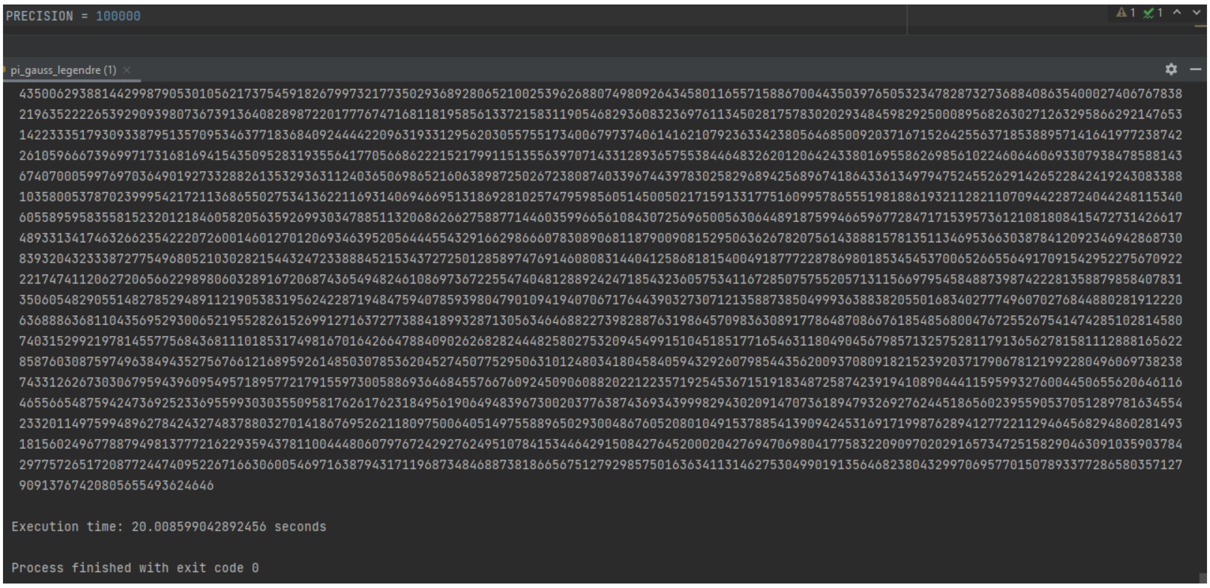
Testes

Primeiro teste - Calculando os primeiros 10000 (dez mil) dígitos de π.



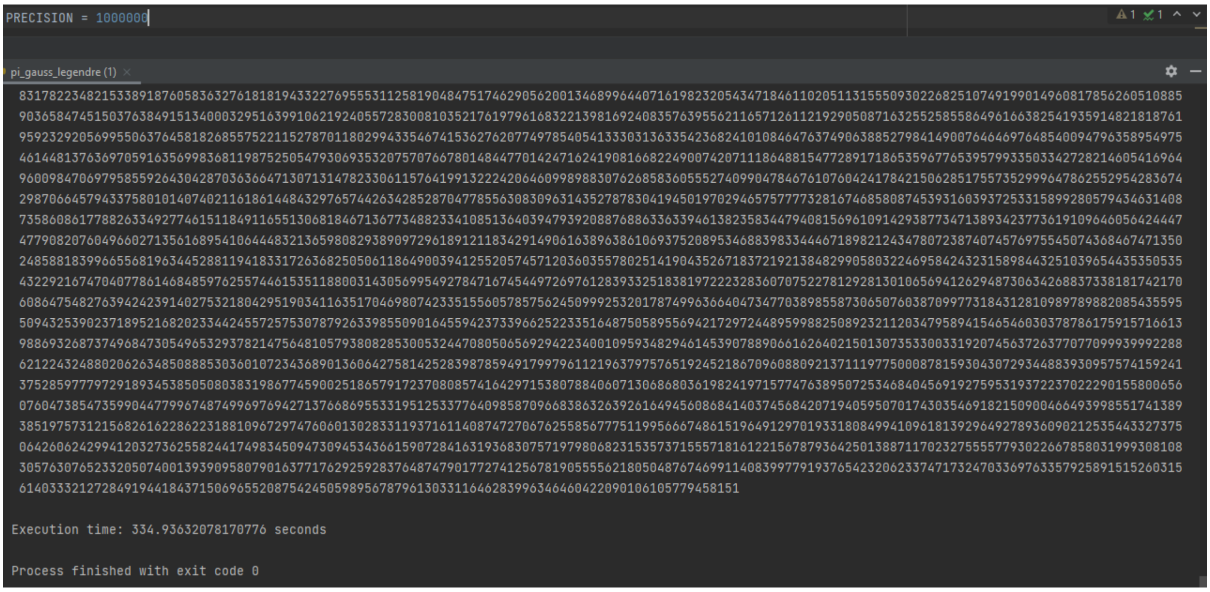
Foram necessários aproximadamente 0.83 segundos.

Segundo teste - Calculando os primeiros 100000 (cem mil) dígitos de π.



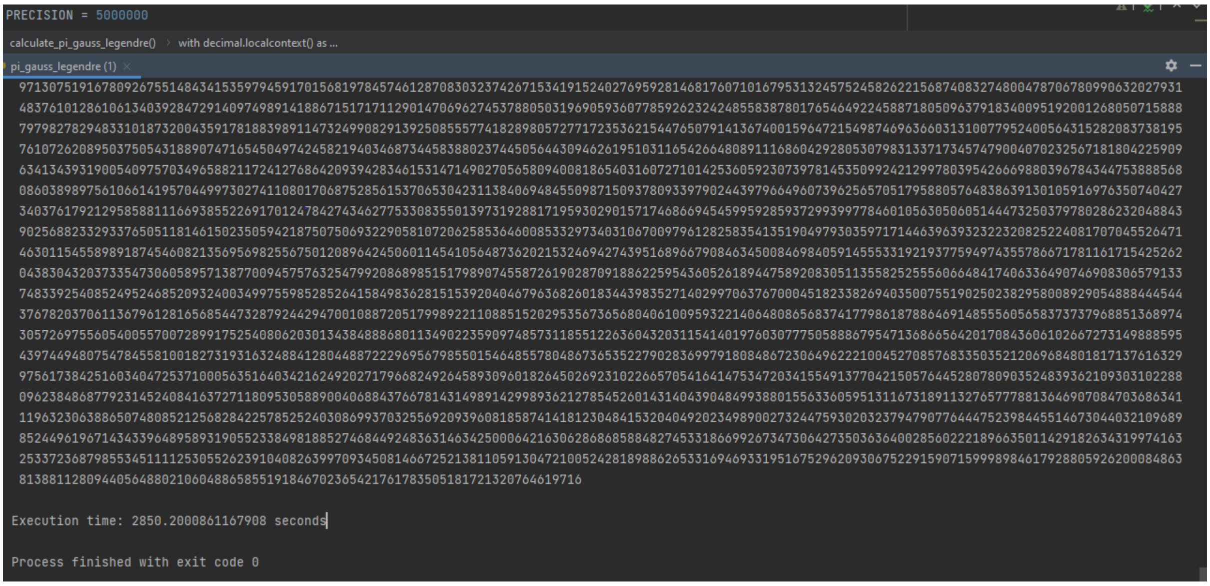
Foram necessários aproximadamente 20 segundos.

Terceiro teste - Calculando os primeiros 1000000 (um milhão) de dígitos de π.



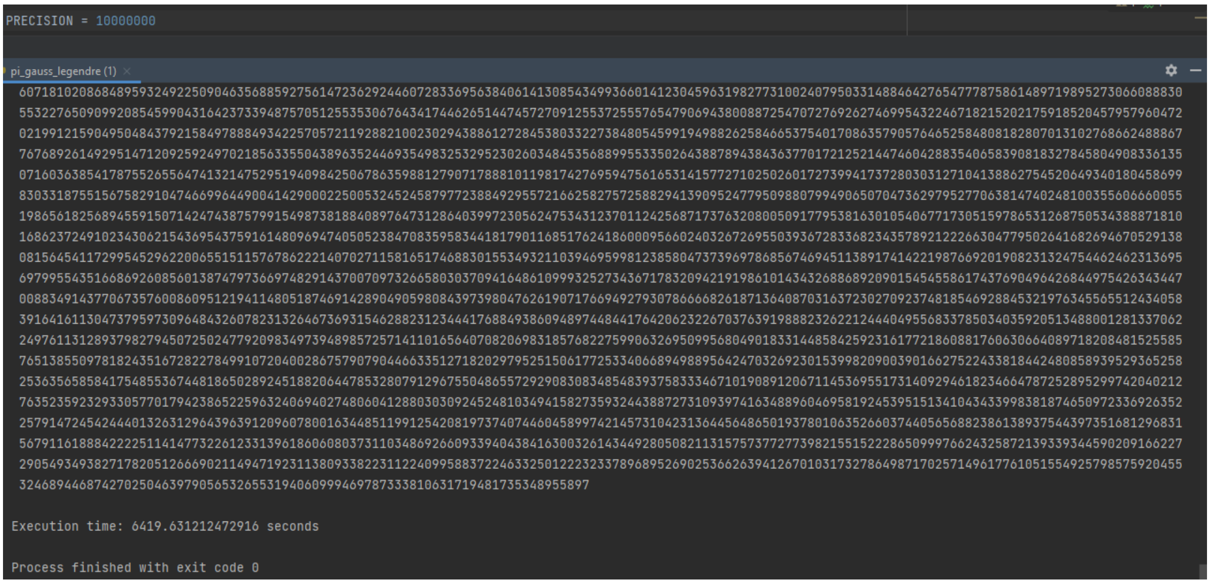
Foram necessários aproximadamente 335 segundos que equivale a 5 minutos e 35 segundos.

Quarto teste - Calculando os primeiros 5000000 (cinco milhões) de dígitos de π.



Foram necessários aproximadamente 2850 segundos que equivale a 47 minutos e 30 segundos.

Quinto teste - Calculando os primeiros 10000000 (dez milhões) de dígitos de π.



Foram necessários aproximadamente 6420 segundos que equivale a 1 hora e 47 minutos.

Os testes acima foram realizados em um computador com processador Intel(R) Core(TM) i5-8250 U CPU @ 1.60GHz 1.80 GHz e 8GB de memória RAM.

Outros algoritmos mais rápidos que a série de Leibniz

Algoritmo Chudnovsky

Método rápido para calcular os dígitos de π, com base na fórmula produzida por Ramanujan. Esse algoritmo foi publicado em 1988 pelos irmãos Chudnovsky.

$$\frac{1}{\pi} = 12\sum_{q=0}^{\infty} \frac{(-1)^q (6q)!(545140134q + 13591409)}{(3q)!(q!)^3 (640320)^{3q+3/2}}$$

Foi utilizada para quebrar o recorde mundial calculando 2.7 trilhões de dígitos de π em dezembro de 2009, 10 trilhões de dígitos em outubro de 2011, 22.4 trilhões de dígitos em novembro de 2016, 31.4 trilhões de dígitos em setembro de 2018 - janeiro de 2019.

A quebra de recorde de 31.4 trilhões de dígitos foi realizada por Emma Haruka, cientista da computação da Google. O processo demorou cerca de 121 dias, a Google utilizou 96 CPUs virtuais com 1.4 TB de memória RAM, leu um total de 9.02 petabytes em dados e escreveu outros 7.95 petabytes. A Google Cloud publicou um artigo muito interessante relacionado à conquista.

Algoritmo BBP

O algoritmo "Bailey-Borwein-Plouffe" (BBP) para cálculo de π é baseado na fórmula BBP para cálculo de π, que foi descoberta em 1995 e publicada em 1996.

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right).$$

Esta fórmula permite que π seja calculado rapidamente para qualquer dada precisão. Um fato interessante é que a fórmula BBP foi utilizada de maneira independente para verificar a corretude dos dígitos de π após a quebra do recorde mundial em 2019.

Referências

https://en.wikipedia.org/wiki/Leibniz_formula_for_%CF%80
https://en.wikipedia.org/wiki/Chudnovsky_algorithm
https://en.wikipedia.org/wiki/Gauss%E2%80%93Legendre_algorithm
https://www.piday.org/million/
https://www.researchgate.net/publication/228702113_The_BBP_Algorithm_for_Pi
https://cloud.google.com/blog/products/compute/calculating-31-4-trillion-digits-of-archimedes-constant-on-google-cloud