CÁLCULO DO PI PELO MÉTODO DE MONTE CARLO UTILIZANDO PYTHON

A atividade propunha na resolução do cálculo do PI pelo método de Monte Carlo, onde para esta resolução fora utilizada a linguagem Python.

O método de Monte Carlo utiliza-se de dois valores (X e Y) gerados aleatoriamente entre 0 e 1, esses valores são utilizados para o cálculo entre a distância entre X e Y aplicando o teorema de Pitágoras.

Caso a soma das distâncias de X e Y forem menores que o raio (1), diz-se que é um ponto interior (dentro da circunferência), no entanto se a soma das distâncias de X e Y forem maiores que o raio, diz-se que é um ponto exterior, logo, fora da circunferência. E através desses pontos, dá para se obter o valor de PI, utilizando a fórmula: 4*N/T, onde N é o número total de pontos interiores da circunferência e T a soma dos números totais de pontos interiores e exteriores da circunferência.

Foram utilizadas três bibliotecas para a resolução do problema, sendo elas, matplotlib, random e multiprocessing. A matplotlib com o intuito de plotar os gráficos, a random, nativa da linguagem python, utilizada para gerar números randômicos, e a multiprocessing, utilizadas para setar e manipular threads no código.

Iniciado a resolução do problema com a criação de uma função pi, onde ela recebe por parâmetro o número total de pontos. Essa função tem como finalidade gerar os números de X e Y aleatoriamente, e calcular se a soma desses números condiz se o ponto está dentro ou fora de circunferência, caso esteja dentro, um contador se auto incrementa.

A função main é a função principal do código, onde são passados o número total de pontos e o números de threads (workers). O total de pontos será dividido igualmente para cada thread, e aplicado a função PI para cada um dos pontos sejam calculados.

Ao final, será plotado um gráfico, contendo todos os pontos em que cada thread realizou o cálculo. Após a aplicação da fórmula e retornado a estimativa do valor de PI.

Abaixo, segue as imagens referente a resolução do projeto e também o plot do gráfico contendo todos os pontos e o valor aproximado de PI.

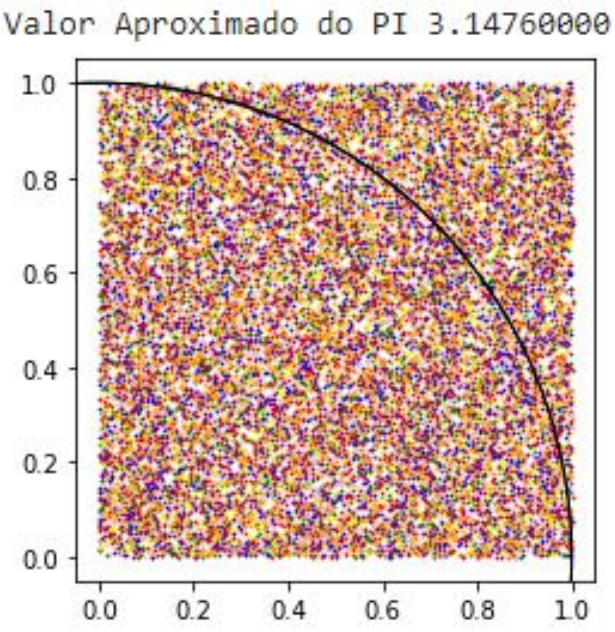

Figura 1 - Plot de todos os pontos em suas Respectivas Threads.

Figura 2 - Legenda Para Cores das Threads

```
[56] import random

     def pi(n):
         count = 0
         insideX = []
         insideY = []
         outsideX = []
         outsideY = []
         for i in range(n):
             x = random.uniform(0, 1)
             y = random.uniform(0, 1)
             if (x ** 2 + y ** 2) < 1.0:
                 count = count + 1
                 insideX.append(x)
                 insideY.append(y)
             else:
                 outsideX.append(x)
                 outsideY.append(y)
         return (count, insideX, insideY, outsideX, outsideY)
```

Figura 3 - Código referente a função PI

```
import multiprocessing
import matplotlib.pyplot as plt
if __name__ == "__main__":
  threads = 8;
  total = 20000
  count, insideX, insideY, outsideX, outsideY = pi(total)
  threadsTotal = []
  for i in range(threads):
    calc = int(total/threads)
    threadsTotal.append(calc)
  pool = multiprocessing.Pool(threads)
  result = pool.map(pi, threadsTotal)

fig, ax = plt.subplots()
circle = plt.Circle((0, 0), 1, fill=False)
ax.set_aspect(1)
ax.add_artist(circle)
countColor = 0
colorThreads = ['red', 'green', 'blue', 'yellow', 'pink', 'brown', 'purple', 'orange']

for tupla in result:
    inX = tupla[1]
    inY = tupla[2]
    outX = tupla[3]
    outY = tupla[4]
    ax.set_aspect('equal')
    ax.scatter(inX, inY, color=colorThreads[countColor], s=0.5)
    ax.scatter(outX, outY, color=colorThreads[countColor], s=0.5)
    countColor = countColor + 1


fig.savefig('PI.pdf')
piValue = float(count*4)/total
print('Valor Aproximado do PI {:.8f}'.format(piValue))
```

Figura 4 - Código Fonte Função Main