Unidade IV - Probabilidade e Estatística

Grupo : Wanderson Faustino Patricio

João Isaac Alves Farias

Karla Mikaelly Paz de Almeida

Durante várias etapas da programação é necessário executar buscas em bancos de dados. Existem vários algoritmos de busca, cada um tenho um desempenho específico e um tempo de execução característico, que pode variar dependendo do sistema operacional, da memória do computador utilizado para executar o algoritmo, entre outras coisas.

Dentre os parâmetros que podem alterar o tempo de execução de uma busca está o tamanho da entrada fornecida, ou seja, a quantidade de elementos que deverão ser visitados até ser encontrado o elemento desejado.

Para esse estudo analisaremos tempo de execução de um algoritmo de busca linear em uma lista não ordenada. Como o funcionamento do algoritmo baseia-se em visitar todos os elementos da lista um a um em sequência até encontrar o elemento chave, caso a lista não possua o número tido como parâmetro da procura, o programa terá que visitar todos os elementos da lista até terminar sua execução. Desta forma, considerando que a visita a um elemento seja realizada em um tempo constante, o tempo de execução total do algoritmo dependerá linearmente do tamanho da lista.

Escrevendo em notação "big O" temos:

$$T(n) \in O(n) \Rightarrow T(n) \approx A + B \cdot n$$

Onde $T(n)$ é o tempo que o algoritmo leva para varrer uma lista de tamanho n.

Para minimizar possíveis erros devido a outros fatores além do tamanho da entrada faremos o valor de n "varrer valores grandes" $\left(n \geq 10^6\right)$.

Executando o algoritmo e registrando os resultados montamos a seguinte tabela:

Adicionando os dados coletados em uma lista.

```
# Declarando os tamanhos da entrada
n = [i*1000000 for i in range(1, 41)]
nReduzido = [i for i in range(1, 41)]

# Declarando os tempos de acordo com a tabela
tempo = [0.885, 0.9041, 0.9262, 0.9489, 0.971, 0.9768, 0.994, 1.021, 1.031,
         1.053, 1.087, 1.082, 1.11, 1.121, 1.151, 1.175, 1.186, 1.223, 1.236,
         1.248, 1.268, 1.284, 1.313, 1.32, 1.341, 1.357, 1.376, 1.402, 1.402,
         1.443, 1.462, 1.477, 1.497, 1.502, 1.511, 1.536, 1.547, 1.564, 1.611,
         1.608]
```

Fazendo o gráfico de dispersão dos pontos $tempo$ x n:

```
import pandas as pd

dataFrame = pd.DataFrame({'n (em milhões)' : nReduzido, 'Tempo (s)' : tempo},
                         index = ['' for i in range(40)])

print(dataFrame)
```

```
       n (em milhões)  Tempo (s)
                    1     0.8850
                    2     0.9041
                    3     0.9262
                    4     0.9489
                    5     0.9710
                    6     0.9768
                    7     0.9940
                    8     1.0210
                    9     1.0310
                   10     1.0530
                   11     1.0870
                   12     1.0820
                   13     1.1100
                   14     1.1210
                   15     1.1510
                   16     1.1750
                   17     1.1860
                   18     1.2230
                   19     1.2360
                   20     1.2480
                   21     1.2680
                   22     1.2840
                   23     1.3130
```

```
24     1.3200
25     1.3410
26     1.3570
27     1.3760
28     1.4020
29     1.4020
30     1.4430
31     1.4620
32     1.4770
33     1.4970
34     1.5020
35     1.5110
36     1.5360
37     1.5470
38     1.5640
39     1.6110
40     1.6080
```

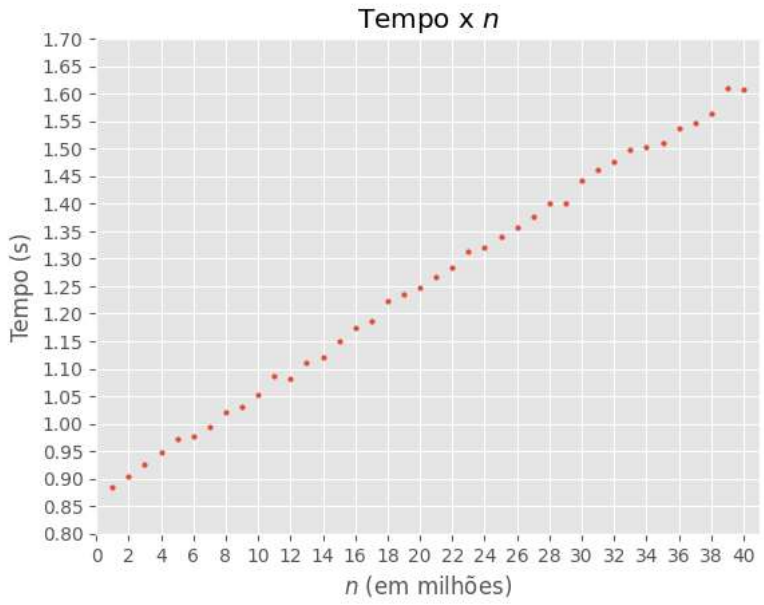
```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('ggplot')
fig, ax = plt.subplots()

ax.scatter(nReduzido, tempo, s = 6)
ax.set_title('Tempo x $n$')
ax.set_xlabel('$n$ (em milhões)')
ax.set_ylabel('Tempo (s)')

ax.set(xlim = (0, 41), ylim = (0.8, 1.7), xticks = np.linspace(0, 40, 21), yticks = np.linspace(0.8, 1.7, 19))

plt.show()
```



Definindo o tamanho n da entrada como variável x e o tempo de execução do programa como variável y podemos calcular o coeficiente de correlação de Pearson através da fórmula:

$$r = \frac{n \cdot \sum(xy) - \left(\sum x\right) \cdot \left(\sum y\right)}{\sqrt{n \cdot \left(\sum x^2\right) - \left(\sum x\right)^2} \cdot \sqrt{n \cdot \left(\sum y^2\right) - \left(\sum y\right)^2}}$$

Definiremos algumas funções para o cálculo de r que recebem como parâmetros listas.

$$f_1(lista) = \sum x^2$$

```
def f1(lista):
    soma = 0
    for x in lista:
        soma += x**2

    return soma
```

$$f_2(lista) = \sum x$$

```
def f2(lista):
    soma = 0
    for x in lista:
        soma += x

    return soma
```

$$f_3(lista01, lista02) = \sum(xy)$$

```
def f3(lista01, lista02):
    soma = 0
    for i in range(len(lista01)):
        soma += lista01[i]*lista02[i]

    return soma
```

O coeficiente de correlação ficará:

$$r = \frac{n \cdot f_3(x,y) - f_2(x) \cdot f_2(y)}{\sqrt{n \cdot f_1(x) - [f_2(x)]^2} \cdot \sqrt{n \cdot f_1(y) - [f_2(y)]^2}}$$

```
from math import sqrt

def r(lista01, lista02):
    n = len(lista01)

    r = (n*f3(lista01, lista02)-f2(lista01)*f2(lista02))/(sqrt(n*f1(lista01)-f2(lista01)**2)*sqrt(n*f1(lista02)-f2(lista02)**2))

    return r
```

Calculando o r para as listas n e $tempo$.

```
print('Coeficiente de correlação: r = {:.5f}'.format(r(n, tempo)))
```

```
Coeficiente de correlação: r = 0.99921
```

Como $r = 0,9921$ (muito próximo de 1) temos uma forte indicação de que há uma dependência linear entre o tempo de execução e o tamanho da entrada. Porém, é necessário fazer um teste de hipótese para ter uma indicação mais forte.