



**Aluna:** Daniela América da Silva  
**Disciplina:** CT208 - Matemática Computacional  
**Prof. Ney Soma**  
**Orientação:** Prof. Tasinaffo  
**Co-orientação:** Prof. Johnny  
**Data:** 09/Junho/2020

## **Exercício: Maior Elemento (Resultado Final)**

### **1- Introdução**

Este relatório apresenta os quatro versionamentos do programa para o cálculo de trocas do maior, menor, segundo maior e segundo menor elemento em uma lista criada a partir da permutação dos números de 1 a 15 [3].

A primeira versão do programa apresenta o processamento de trocas para a lista de permutação do número 1 ao 11, e houve falta de memória para o cálculo do número 11 ao 15.

A segunda versão possibilitou o processamento até o número 15 utilizando o cálculo por recorrências e houve pouca utilização de memória. O programa possui dois passos: (i) armazenamento das trocas para a lista de permutação de 1 até 10; (ii) cálculo das trocas de 11 a 15 a partir das recorrências.

A terceira versão do programa calcula adicionalmente o número de trocas para o menor número e também para o segundo maior e o segundo menor para a lista de permutações do número 1 ao 15. Não houve impactos na performance, uma vez que encontrar o menor significa apenas mais uma cláusula *if* na função que conta as trocas. E encontrar o segundo maior ou segundo menor, é apenas uma atribuição de duas variáveis adicionais, pois ao encontrar o maior significa que o valor anterior é o segundo maior, e ao encontrar o menor significa que o valor anterior é o segundo menor respectivamente.

A quarta versão do programa, utiliza *divide and conquer* [4][7] para o cálculo do maior/segundo maior e menor/segundo menor para os números de 1 à 10. Observa-se que o número de trocas corresponde ao fatorial de  $n$ . Dito isto dos números de 11 a 15 é realizado o cálculo do fatorial através do memoization [4][5][6][8]. Observa-se que a soma dos valores de cada linha do método anterior também corresponde ao fatorial.

O relatório contém também a demonstração matemática apenas para a notação Big O. As funções geratrizes serão demonstradas em aula [1][2].

Para o estudo o código foi desenvolvido utilizando a linguagem Python e o Google Colab para execução do programa. As bibliotecas Python utilizadas são *itertools* para cálculo das permutações e *numpy* para utilização de matriz para o armazenamento dos dados. O cálculo das trocas, bem como o cálculo por recorrências, fatorial e uso de *divide and conquer* foi codificado sem uso de bibliotecas.

O código desenvolvido em Python está disponível no Github  
[https://github.com/dasamerica/ct208/tree/master/aula\\_2805\\_maior\\_elemento](https://github.com/dasamerica/ct208/tree/master/aula_2805_maior_elemento)

## 2- Simulador Python

Execução realizada no Google Colab pois propicia até 12Gb de memória.

A permutação para a geração da lista de elementos é realizada via biblioteca itertools.

A função que conta as trocas até determinar o maior elemento não utiliza biblioteca phyton.

É possível executar o programa de 1 até 10 elementos em uma única execução que consome até 1Gb de memória RAM.

Para 11 elementos é necessário a execução individual consumindo até 7 Gb memória RAM.

Para 12 e 13 elementos não há memória suficiente.

Código Fonte:

[https://github.com/dasamerica/ct208/blob/master/aula\\_2805\\_maior\\_elemento/ct208\\_maior\\_elemento.py](https://github.com/dasamerica/ct208/blob/master/aula_2805_maior_elemento/ct208_maior_elemento.py)

Programa Principal	Funções
<pre># Programa phyton para dado um numero gera uma lista de suas permutações # Verifica quantas iterações necessárias para achar o maior elemento da lista # Daniela América da Silva # Disciplina: ct208  #processa lista de permutações do número 1 ao 10 x = 1 while x &lt; 11:     # inicializa lista de trocas     trocas = []      numeros = cria_lista (x)      #inicializa contador trocas     conta_trocas = set_contador(len(numeros))      # realiza as permutações     perm = permutations(numeros)      # Conta as trocas     for i in list(perm):         #print (i)         number = maior_trocas (i)         #imprime</pre>	<pre># Programa phyton para dado um numero gera uma lista de suas permutações # Verifica quantas iterações necessárias para achar o maior elemento da lista # Daniela América da Silva # Disciplina: ct208  # Importa library de permutações from itertools import permutations  #seta vetor contador de trocas def set_contador (col):     n = 0     conta = []     while n &lt; col:         conta.append(0)         n+=1     return (conta)  #cria a lista de números def cria_lista (num):     n = 0     lista = []     while n &lt; num:         lista.append(n+1)         n+=1     return (lista)</pre>

```

    #print (i, '(', number ,')')
    #trocas.append(number)
    conta_trocas[number]+=1

    #print (trocas)
    print ('numero:', x, conta_trocas)
    x+=1

```

```

#conta numero de trocas
def maior_trocas (elements):
    m = elements[0]
    n = 1
    t = 0
    while n < (len(elements)):
        if m < elements[n]:
            m = elements[n]
            t+=1
        n+=1
    return (t)

```

Resultados para lista de permutações até 10 elementos:

```

numero: 1 [1]
numero: 2 [1, 1]
numero: 3 [2, 3, 1]
numero: 4 [6, 11, 6, 1]
numero: 5 [24, 50, 35, 10, 1]
numero: 6 [120, 274, 225, 85, 15, 1]
numero: 7 [720, 1764, 1624, 735, 175, 21, 1]
numero: 8 [5040, 13068, 13132, 6769, 1960, 322, 28, 1]
numero: 9 [40320, 109584, 118124, 67284, 22449, 4536, 546, 36, 1]
numero: 10 [362880, 1026576, 1172700, 723680, 269325, 63273, 9450, 870, 45, 1]

```

The screenshot shows a Jupyter Notebook titled 'ct208\_maior\_elemento.ipynb'. The code cell contains the same Python code as shown in the previous blocks. The output cell displays the results for numbers 1 through 10, matching the text provided. The interface includes a top bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus. On the right, there are buttons for 'Comment', 'Share', and a user profile icon. Below the code cell, a status bar indicates 'Connected to "Python 3 Google Compute Engine backend"' and shows memory usage: 'RAM: 0.81 GB/12.72 GB Disk: 31.03 GB/107.77 GB'.

Resultados para lista de permutações com 11 elementos, aproximadamente 7Gb memória

**\*\* reduziu o loop para apenas 11 elementos**

```

#processa lista de permutações do número 1 ao 10
x = 11
while x < 12:
    # inicializa lista de trocas
    trocas = []

```

numero: 11 [3628800, 10628640, 12753576, 8409500, 3416930, 902055, 157773, 18150, 1320, 55, 1]

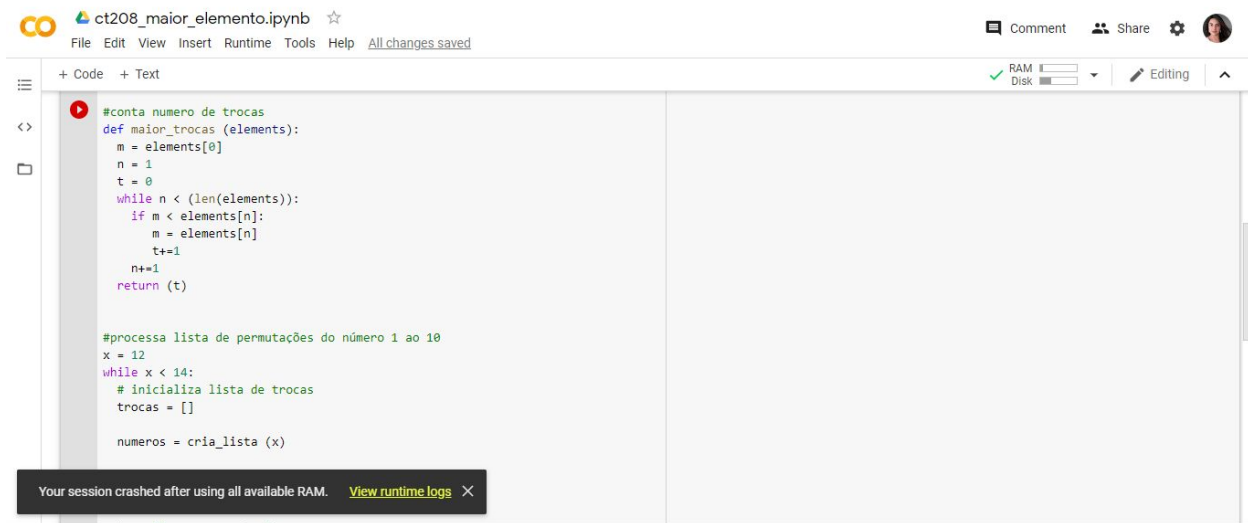


```
for i in range(perm):
    #print (i)
    number = maior_trocas (i)
    #imprime
    #print (i, '(', number, ')')
    #trocas.append(number)
    conta_trocas[number]+=1

#print (trocas)
print ('numero:', x, conta_trocas)
x+=1
```

numero: 11 [3628800, 10628640, 12753576, 8409500, 3416930, 902055, 157773, 18150, 1320, 55, 1]

Resultados para lista de permutações com 12 ou 13 elementos, não há memória suficiente, acima de 12 Gb necessários.



```
#conta numero de trocas
def maior_trocas (elements):
    m = elements[0]
    n = 1
    t = 0
    while n < (len(elements)):
        if m < elements[n]:
            m = elements[n]
            t+=1
        n+=1
    return (t)

#processa lista de permutações do número 1 ao 10
x = 12
while x < 14:
    # inicializa lista de trocas
    trocas = []

    numeros = cria_lista (x)
```

Your session crashed after using all available RAM. [View runtime logs](#)

### 3 - Definição de Média Aritmética e Variância

Média aritmética é definida como a “somatório de todos os elementos da série divididos pelo número de elementos”.

$$\text{Média aritmética} = \frac{\sum_{i=1}^n X_i}{n}$$

Variância é definida como a “média dos quadrados das diferenças, entre os valores em relação à sua própria média”.

$$\text{Variância} = \sum_{i=1}^n (X_i - \bar{X})^2$$

$$\frac{\quad}{n - 1}$$

Para a tabela 3, a média e a variância são apresentadas a seguir.

	Qtd trocas	$(X_i - \bar{X})^2$
1 2 3	2	1.361111111
1 3 2	1	0.02777777778
2 1 3	1	0.02777777778
2 3 1	1	0.02777777778
3 1 2	0	0.6944444444
3 2 1	0	0.6944444444
	0.8333333333	0.5666666667
	Média	Variância

#### 4- Programa recursivo $T_{n,k}$

Por ser possível executar o programa de 1 até 10 elementos em uma única execução que consome até 1Gb de memória RAM, os valores foram salvos em uma matriz, e a partir desta matriz foram calculadas os valores através da recorrência  $T_{n,k}$

$$T_{n,k} = T_{n-1,k-1} + (n-1) T_{n-1,k}$$

A utilização de recorrências é linear em n, pois o número de células da matriz de recorrência que armazenam os valores das trocas serão lidas de forma linear n. O valor de k é no máximo n, pois corresponde ao número de células que serão atualizadas.

Código Fonte:

[https://github.com/dasamerica/ct208/blob/master/aula\\_2805\\_maior\\_elemento/ct208\\_maior\\_elemento\\_recursivo.py](https://github.com/dasamerica/ct208/blob/master/aula_2805_maior_elemento/ct208_maior_elemento_recursivo.py)

Programa Principal	Funções
<pre># Programa phyton para dado um numero gera uma lista de suas permutações # Verifica quantas iterações necessárias para achar o maior elemento da lista # Daniela América da Silva # Disciplina: ct208</pre>	<pre># Programa phyton para dado um numero gera uma lista de suas permutações # Verifica quantas iterações necessárias para achar o maior elemento da lista # Daniela América da Silva # Disciplina: ct208</pre>

```

# inicializa matriz t que armazena resultados
das trocas
t = np.zeros((15, 16))
t[0][0] = 1

#processa lista de permutações do número 2 ao
10
x = 2
while x < 11:
    # inicializa lista de trocas
    trocas = []

    numeros = cria_lista (x)

    #inicializa contador trocas
    conta_trocas = set_contador(len(numeros))

    # realiza as permutações
    perm = permutations(numeros)

    # Conta as trocas
    for i in list(perm):
        #print (i)
        number = maior_trocas (i)
        #imprime
        #print (i, '(', number ,')')
        conta_trocas[number]+=1

    #print (trocas)
    #print ('numero:', x, conta_trocas)

    #atualiza matriz trocas
    j = 0
    t[x-1][j]=x
    j = 1
    while j < x+1:
        t[x-1][j] = conta_trocas[j-1]
        j+=1

    x+=1

#np.set_printoptions(precision=10,
suppress=True, linewidth=10000)

np.set_printoptions(formatter=('float':
lambda x: ' ' + str(x)})

```

```

# Importa library de permutações
from itertools import permutations

import numpy as np

#seta vetor contador de trocas
def set_contador (col):
    n = 0
    conta = []
    while n < col:
        conta.append(0)
        n+=1
    return (conta)

#cria a lista de números
def cria_lista (num):
    n = 0
    lista = []
    while n < num:
        lista.append(n+1)
        n+=1
    return (lista)

#conta numero de trocas
def maior_trocas (elements):
    m = elements[0]
    n = 1
    t = 0
    while n < (len(elements)):
        if m < elements[n]:
            m = elements[n]
            t+=1
        n+=1
    return (t)

```

```

#processa lista de permutações do número 11
ao 15, utilizando a recorrência
l=10
c=1
while l < 15:
    t[l][0] = l+1
    t[l][c] = l*t[l-1][c]
    while c < l:
        c+=1
        t[l][c] = t[l-1][c-1] + l*t[l-1][c]
    c+=1
    t[l][c] = t[l-1][c-1] + l*t[l-1][c]
    l+=1
    c=1

print (t)

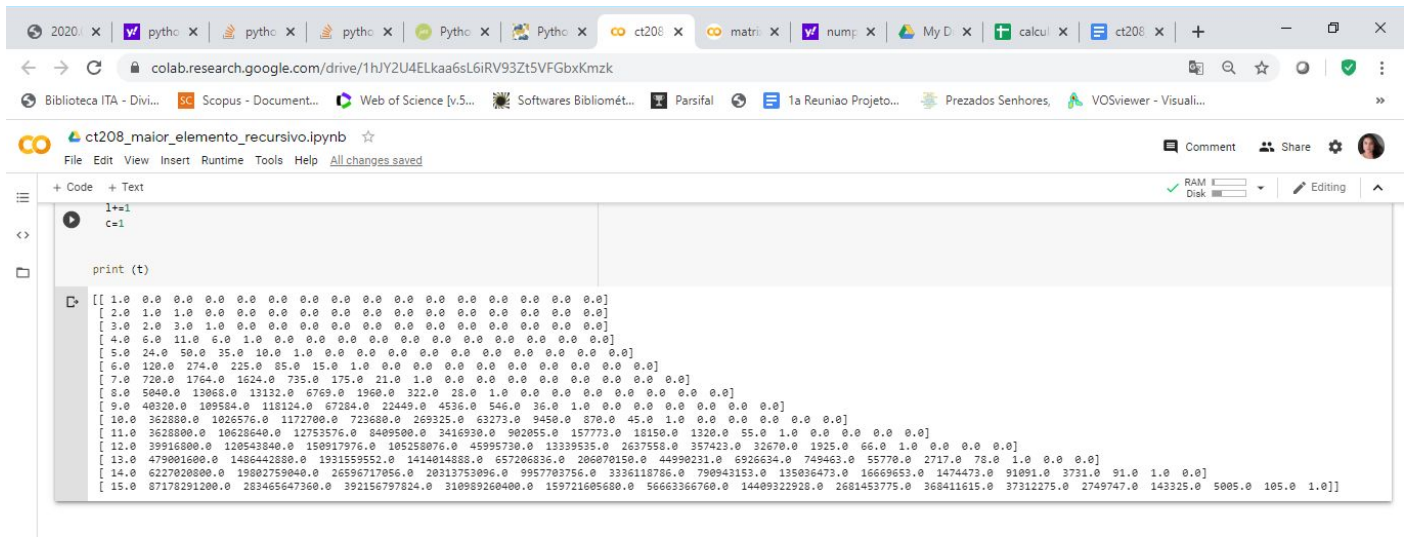
```

## Resultados até 15 elementos

```

[[ 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 2.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 3.0  2.0  3.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 4.0  6.0 11.0  6.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 5.0 24.0 50.0 35.0 10.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 6.0 120.0 274.0 225.0 85.0 15.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 7.0 720.0 1764.0 1624.0 735.0 175.0 21.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 8.0 5040.0 13068.0 13132.0 6769.0 1960.0 322.0 28.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 9.0 40320.0 109584.0 118124.0 67284.0 22449.0 4536.0 546.0 36.0  1.0  0.0  0.0  0.0  0.0]
 [10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 63273.0 9450.0 870.0 45.0  1.0  0.0  0.0  0.0]
 [11.0 3628800.0 10628640.0 12753576.0 8409500.0 3416930.0 902055.0 157773.0 18150.0 1320.0 55.0  1.0  0.0  0.0]
 [12.0 39916800.0 120543840.0 150917976.0 105258076.0 45995730.0 13339535.0 2637558.0 357423.0 32670.0 1925.0 66.0  1.0  0.0]
 [13.0 479001600.0 1486442880.0 1931559552.0 1414014888.0 657206836.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0  1.0]
 [14.0 6227020800.0 19802759040.0 26596717056.0 20313753096.0 9957703756.0 3336118786.0 790943153.0 135036473.0 16669653.0 1474473.0 91091.0 3731.0 91.0  1.0]
 [15.0 87178291200.0 283465647360.0 392156797824.0 310989260400.0 159721605680.0 56663366760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 143325.0 5005.0 105.0  1.0]]

```



### 5- Cálculo do maior, menor, segundo maior e segundo menor

Não há alteração na performance, e no número de trocas, uma vez que encontrar o menor elemento significa apenas mais um *if* na função que calcula as trocas, e encontrar o segundo maior ou segundo menor elemento, é apenas uma atribuição de duas novas variáveis. Ou seja, ao encontrar o maior elemento significa que o valor anterior é o segundo maior elemento, e ao encontrar o menor elemento significa que o valor anterior é o segundo menor elemento. Portanto conforme demonstrado na execução do programa, não há alteração de performance ou aumento no número de trocas com estas extensões no programa.

Segue o código para cálculo do maior, menor, segundo maior e segundo menor elementos com a demonstração do mesmo número de troca.

Código fonte:

Apenas maior e menor elemento:

[https://github.com/dasamerica/ct208/blob/master/aula\\_2805\\_maior\\_elemento/ct208\\_maior\\_menor\\_elemento\\_recurso.py](https://github.com/dasamerica/ct208/blob/master/aula_2805_maior_elemento/ct208_maior_menor_elemento_recurso.py)

Com maior/menor/segundo maior/segundo menor elemento:

[https://github.com/dasamerica/ct208/blob/master/aula\\_2805\\_maior\\_elemento/ct208\\_maior\\_men](https://github.com/dasamerica/ct208/blob/master/aula_2805_maior_elemento/ct208_maior_men)  
or segundo elemento recursivo.py

Programa Principal	Funções
<pre># Programa phyton para dado um numero gera uma lista de suas permutações # Verifica quantas iterações necessárias para achar o maior, menor, segundo maior e segundo menor elemento da lista # Daniela América da Silva # Disciplina: ct208</pre>	<pre># Programa phyton para dado um numero gera uma lista de suas permutações # Verifica quantas iterações necessárias para achar o maior, menor, segundo maior e segundo menor elemento da lista # Daniela América da Silva # Disciplina: ct208</pre>



```

# inicializa matriz t que armazena resultados
das trocas
t_maior = np.zeros((15, 16))
t_maior[0][0] = 1

t_menor = np.zeros((15, 16))
t_menor[0][0] = 1

t_maior_segundo = np.zeros((15, 16))
t_maior_segundo[0][0] = 1

t_menor_segundo = np.zeros((15, 16))
t_menor_segundo[0][0] = 1

#processa lista de permutações do número 2 ao
10
x = 2
while x < 11:
    # inicializa lista de trocas
    trocas = []

    numeros = cria_lista (x)

    #inicializa contador trocas
    conta_trocas_maior =
set_contador(len(numeros))
    conta_trocas_menor =
set_contador(len(numeros))
    conta_trocas_maior_segundo =
set_contador(len(numeros))
    conta_trocas_menor_segundo =
set_contador(len(numeros))

    # realiza as permutações
    perm = permutations(numeros)

    # Conta as trocas
    for i in list(perm):
        #print (i)
        number = maior_menor_trocas (i)
        #imprime
        #print (i, '(', number ,')')
        conta_trocas_maior[number[0]]+=1
        conta_trocas_menor[number[1]]+=1
        conta_trocas_maior_segundo[number[2]]+=1
        conta_trocas_menor_segundo[number[3]]+=1

```

```

# Importa library de permutações
from itertools import permutations

import numpy as np

#seta vetor contador de trocas
def set_contador (col):
    n = 0
    conta = []
    while n < col:
        conta.append(0)
        n+=1
    return (conta)

#cria a lista de números
def cria_lista (num):
    n = 0
    lista = []
    while n < num:
        lista.append(n+1)
        n+=1
    return (lista)

#conta numero de trocas incluindo o segundo
elemento
def maior_menor_trocas (elements):
    me = elements[0]
    ma = elements[0]
    ma_2 = elements[0]
    me_2 = elements[0]
    n = 1
    t= [0,0,0,0]
    while n < (len(elements)):
        # conta trocas maior
        if ma < elements[n]:
            ma_2 = ma
            ma = elements[n]
            t[0]+=1
            t[2]+=1

        # conta trocas menor
        if me > elements [n]:
            me_2 = me
            me = elements[n]
            t[1]+=1
            t[3]+=1

        n+=1
    return (t)

```

```

#print (trocas)
#print ('numero:', x, conta_trocas)

#atualiza matriz trocas
j = 0
t_maior[x-1][j]=x
t_menor[x-1][j]=x
t_maior_segundo[x-1][j]=x
t_menor_segundo[x-1][j]=x

j = 1
while j < x+1:
    t_maior[x-1][j] = conta_trocas_maior[j-1]
    t_menor[x-1][j] = conta_trocas_menor[j-1]
    t_maior_segundo[x-1][j] =
conta_trocas_maior_segundo[j-1]
    t_menor_segundo[x-1][j] =
conta_trocas_menor_segundo[j-1]
    j+=1

x+=1

#np.set_printoptions(precision=10,
suppress=True, linewidth=10000)

np.set_printoptions(formatter={'float':
lambda x: ' ' + str(x)})

#processa lista de permutações do número 11
ao 15, utilizando a recorrência
l=10
c=1
while l < 15:
    t_maior[l][0] = l+1
    t_maior[l][c] = l*t_maior[l-1][c]
    t_menor[l][0] = l+1
    t_menor[l][c] = l*t_menor[l-1][c]
    t_maior_segundo[l][0] = l+1
    t_maior_segundo[l][c] =
l*t_maior_segundo[l-1][c]
    t_menor_segundo[l][0] = l+1
    t_menor_segundo[l][c] =
l*t_menor_segundo[l-1][c]

while c < l:
    c+=1

```

```

        t_maior[l][c] = t_maior[l-1][c-1] +
1*t_maior[l-1][c]
        t_menor[l][c] = t_menor[l-1][c-1] +
1*t_menor[l-1][c]
        t_maior_segundo[l][c] =
t_maior_segundo[l-1][c-1] +
1*t_maior_segundo[l-1][c]
        t_menor_segundo[l][c] =
t_menor_segundo[l-1][c-1] +
1*t_menor_segundo[l-1][c]

        c+=1
        t_maior[l][c] = t_maior[l-1][c-1] +
1*t_maior[l-1][c]
        t_menor[l][c] = t_menor[l-1][c-1] +
1*t_menor[l-1][c]
        t_maior_segundo[l][c] =
t_maior_segundo[l-1][c-1] +
1*t_maior_segundo[l-1][c]
        t_menor_segundo[l][c] =
t_menor_segundo[l-1][c-1] +
1*t_menor_segundo[l-1][c]

        l+=1
        c=1

print ('CONTA TROCAS PARA MAIOR NUMERO')
print (t_maior)
print ('CONTA TROCAS PARA MENOR NUMERO')
print (t_menor)

print ('CONTA TROCAS PARA SEGUNDO MAIOR
NUMERO')
print (t_maior_segundo)
print ('CONTA TROCAS PARA SEGUNDO MENOR
NUMERO')
print (t_menor_segundo)

```

## Resultados maior, menor, segundo maior e segundo menor para até 15 elementos

CONTA TROCAS PARA MAIOR NUMERO

```

[[ 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 2.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 3.0  2.0  3.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 4.0  6.0 11.0  6.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 5.0 24.0 50.0 35.0 10.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 6.0 120.0 274.0 225.0 85.0 15.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 7.0 720.0 1764.0 1624.0 735.0 175.0 21.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
0.0]
 [ 8.0 5040.0 13068.0 13132.0 6769.0 1960.0 322.0 28.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
0.0 0.0]
 [ 9.0 40320.0 109584.0 118124.0 67284.0 22449.0 4536.0 546.0 36.0  1.0  0.0  0.0  0.0  0.0  0.0]
0.0 0.0 0.0]
 [ 10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 63273.0 9450.0 870.0 45.0  1.0  0.0  0.0  0.0  0.0]
0.0 0.0 0.0 0.0 0.0]
 [ 11.0 3628800.0 10628640.0 12753576.0 8409500.0 3416930.0 902055.0 157773.0 18150.0 1320.0 55.0  1.0  0.0  0.0  0.0  0.0]
 [ 12.0 39916800.0 120543840.0 150917976.0 105258076.0 45995730.0 13339535.0 2637558.0 357423.0 32670.0 1925.0 66.0  1.0  0.0  0.0  0.0]
 [ 13.0 479001600.0 1486442880.0 1931559552.0 1414014888.0 657206836.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0  1.0  0.0  0.0]
 [ 14.0 6227020800.0 19802759040.0 26596717056.0 20313753096.0 9957703756.0 3336118786.0 790943153.0 135036473.0 16669653.0 1474473.0 91091.0 3731.0 91.0  1.0  0.0]
 [ 15.0 87178291200.0 283465647360.0 392156797824.0 310989260400.0 159721605680.0 56663366760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 143325.0 5005.0 105.0  1.0]]

```

#### CONTA TROCAS PARA MENOR NUMERO

```

[[ 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 2.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 3.0  2.0  3.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 4.0  6.0 11.0  6.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 5.0 24.0 50.0 35.0 10.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 6.0 120.0 274.0 225.0 85.0 15.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 7.0 720.0 1764.0 1624.0 735.0 175.0 21.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
0.0]
 [ 8.0 5040.0 13068.0 13132.0 6769.0 1960.0 322.0 28.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
0.0 0.0]
 [ 9.0 40320.0 109584.0 118124.0 67284.0 22449.0 4536.0 546.0 36.0  1.0  0.0  0.0  0.0  0.0  0.0]
0.0 0.0 0.0]
 [ 10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 63273.0 9450.0 870.0 45.0  1.0  0.0  0.0  0.0  0.0]
0.0 0.0 0.0 0.0 0.0]
 [ 11.0 3628800.0 10628640.0 12753576.0 8409500.0 3416930.0 902055.0 157773.0 18150.0 1320.0 55.0  1.0  0.0  0.0  0.0]
 [ 12.0 39916800.0 120543840.0 150917976.0 105258076.0 45995730.0 13339535.0 2637558.0 357423.0 32670.0 1925.0 66.0  1.0  0.0  0.0]
 [ 13.0 479001600.0 1486442880.0 1931559552.0 1414014888.0 657206836.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0  1.0  0.0]
 [ 14.0 6227020800.0 19802759040.0 26596717056.0 20313753096.0 9957703756.0 3336118786.0 790943153.0 135036473.0 16669653.0 1474473.0 91091.0 3731.0 91.0  1.0  0.0]
 [ 15.0 87178291200.0 283465647360.0 392156797824.0 310989260400.0 159721605680.0 56663366760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 143325.0 5005.0 105.0  1.0]]

```

#### CONTA TROCAS PARA SEGUNDO MAIOR NUMERO

```

[[ 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 2.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 3.0  2.0  3.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]

```

```

[ 4.0  6.0 11.0  6.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 5.0 24.0 50.0 35.0 10.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 6.0 120.0 274.0 225.0 85.0 15.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 7.0 720.0 1764.0 1624.0 735.0 175.0 21.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
0.0]
[ 8.0 5040.0 13068.0 13132.0 6769.0 1960.0 322.0 28.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
0.0 0.0]
[ 9.0 40320.0 109584.0 118124.0 67284.0 22449.0 4536.0 546.0 36.0  1.0  0.0  0.0  0.0  0.0  0.0]
0.0 0.0 0.0]
[ 10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 63273.0 9450.0 870.0 45.0  1.0  0.0  0.0  0.0  0.0]
0.0 0.0 0.0 0.0]
[ 11.0 3628800.0 10628640.0 12753576.0 8409500.0 3416930.0 902055.0 157773.0 18150.0 1320.0 55.0  1.0  0.0  0.0  0.0  0.0]
[ 12.0 39916800.0 120543840.0 150917976.0 105258076.0 45995730.0 13339535.0 2637558.0 357423.0 32670.0 1925.0 66.0  1.0  0.0  0.0  0.0]
[ 13.0 479001600.0 1486442880.0 1931559552.0 1414014888.0 657206836.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0  1.0  0.0  0.0]
[ 14.0 6227020800.0 19802759040.0 26596717056.0 20313753096.0 9957703756.0 3336118786.0 790943153.0 135036473.0 16669653.0 1474473.0 91091.0 3731.0 91.0  1.0  0.0]
[ 15.0 87178291200.0 283465647360.0 392156797824.0 310989260400.0 159721605680.0 56663366760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 143325.0 5005.0 105.0  1.0]]

```

#### CONTA TROCAS PARA SEGUNDO MENOR NUMERO

```

[[ 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 2.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 3.0  2.0  3.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 4.0  6.0 11.0  6.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 5.0 24.0 50.0 35.0 10.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 6.0 120.0 274.0 225.0 85.0 15.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 7.0 720.0 1764.0 1624.0 735.0 175.0 21.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
0.0]
[ 8.0 5040.0 13068.0 13132.0 6769.0 1960.0 322.0 28.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
0.0 0.0]
[ 9.0 40320.0 109584.0 118124.0 67284.0 22449.0 4536.0 546.0 36.0  1.0  0.0  0.0  0.0  0.0  0.0]
0.0 0.0 0.0]
[ 10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 63273.0 9450.0 870.0 45.0  1.0  0.0  0.0  0.0  0.0]
0.0 0.0 0.0 0.0]
[ 11.0 3628800.0 10628640.0 12753576.0 8409500.0 3416930.0 902055.0 157773.0 18150.0 1320.0 55.0  1.0  0.0  0.0  0.0]
[ 12.0 39916800.0 120543840.0 150917976.0 105258076.0 45995730.0 13339535.0 2637558.0 357423.0 32670.0 1925.0 66.0  1.0  0.0  0.0]
[ 13.0 479001600.0 1486442880.0 1931559552.0 1414014888.0 657206836.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0  1.0  0.0]
[ 14.0 6227020800.0 19802759040.0 26596717056.0 20313753096.0 9957703756.0 3336118786.0 790943153.0 135036473.0 16669653.0 1474473.0 91091.0 3731.0 91.0  1.0]
[ 15.0 87178291200.0 283465647360.0 392156797824.0 310989260400.0 159721605680.0 56663366760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 143325.0 5005.0 105.0  1.0]]

```

**Trocas maior e menor**

+ Code + Text

RAM Disk Editing

```
print(t_menor_segundo)
print('CONTA TROCAS PARA SEGUNDO MENOR NUMERO')
print(t_menor_segundo)
```

```
CONTA TROCAS PARA MAIOR NUMERO
[[ 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 2.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 3.0  2.0  3.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 4.0  5.0 11.0  5.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 5.0 24.0 50.0 35.0 10.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [ 6.0 120.0 274.0 225.0 85.0 15.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [ 7.0 720.0 1764.0 1624.0 735.0 175.0 21.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [ 8.0 5040.0 13068.0 13112.0 6769.0 1960.0 322.0 25.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [ 9.0 40320.0 109584.0 118124.0 67284.0 22449.0 4536.0 546.0 36.0 1.0 0.0 0.0 0.0 0.0 0.0]
 [10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 63273.0 9450.0 870.0 45.0 1.0 0.0 0.0 0.0 0.0]
 [11.0 3628800.0 10628640.0 12753576.0 8409500.0 3416930.0 902055.0 157773.0 18150.0 1320.0 55.0 1.0 0.0 0.0 0.0]
 [12.0 39916800.0 120543840.0 150917976.0 105258076.0 45995730.0 13339535.0 2637550.0 357423.0 32670.0 1925.0 66.0 1.0 0.0 0.0]
 [13.0 479001600.0 1486442880.0 1931559552.0 1414014888.0 657206836.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0 1.0 0.0]
 [14.0 6227020800.0 19802759040.0 26596717056.0 20313753096.0 9957703756.0 3336118786.0 790943153.0 135036473.0 16669653.0 1474473.0 91891.0 3731.0 91.0 1.0]
 [15.0 87178291200.0 283465647360.0 392156797824.0 310989260400.0 159721605600.0 56663566760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 143325.0 5005.0 105.0]]

CONTA TROCAS PARA MENOR NUMERO
[[ 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 2.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 3.0  2.0  3.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 4.0  5.0 11.0  5.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 5.0 24.0 50.0 35.0 10.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [ 6.0 120.0 274.0 225.0 85.0 15.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [ 7.0 720.0 1764.0 1624.0 735.0 175.0 21.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [ 8.0 5040.0 13068.0 13112.0 6769.0 1960.0 322.0 25.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [ 9.0 40320.0 109584.0 118124.0 67284.0 22449.0 4536.0 546.0 36.0 1.0 0.0 0.0 0.0 0.0]
 [10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 63273.0 9450.0 870.0 45.0 1.0 0.0 0.0 0.0 0.0]
 [11.0 3628800.0 10628640.0 12753576.0 8409500.0 3416930.0 902055.0 157773.0 18150.0 1320.0 55.0 1.0 0.0 0.0 0.0]
 [12.0 39916800.0 120543840.0 150917976.0 105258076.0 45995730.0 13339535.0 2637550.0 357423.0 32670.0 1925.0 66.0 1.0 0.0 0.0]
 [13.0 479001600.0 1486442880.0 1931559552.0 1414014888.0 657206836.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0 1.0 0.0]
 [14.0 6227020800.0 19802759040.0 26596717056.0 20313753096.0 9957703756.0 3336118786.0 790943153.0 135036473.0 16669653.0 1474473.0 91891.0 3731.0 91.0 1.0]
 [15.0 87178291200.0 283465647360.0 392156797824.0 310989260400.0 159721605600.0 56663566760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 143325.0 5005.0 105.0]]

print('CONTA TROCAS PARA SEGUNDO MAIOR NUMERO')
```

### Trocas segundo maior e segundo menor

```
ct208_maior_menor_segundo_elemento_recursivo.ipynb
File Edit View Insert Runtime Tools Help
All changes saved

+ Code + Text

[15]: [15] 717021200: 28345647360: 39215679324: 31098266400: 139712095800: 56663366760: 14409322928: 2681453775: 368411615: 37312275: 2749747: 1432525: 5005: 105: 1.0]]

COUNTACAS PARA SEGUINDO MAIOR NUMERO
[[1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [2.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [3.0 2.0 3.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [4.0 3.0 11.0 6.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [5.0 24.0 50.0 35.0 10.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [6.0 120.0 274.0 225.0 85.0 15.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [7.0 720.0 1764.0 1624.0 735.0 175.0 21.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [8.0 5040.0 13068.0 11312.0 6769.0 1960.0 322.0 28.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [9.0 40320.0 109584.0 110124.0 67284.0 22449.0 4536.0 546.0 36.0 1.0 0.0 0.0 0.0 0.0 0.0]
 [10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 62373.0 9450.0 870.0 45.0 1.0 0.0 0.0 0.0 0.0]
 [11.0 3628800.0 10622640.0 12735376.0 9409500.0 3410930.0 902085.0 157773.0 10150.0 1320.0 55.0 1.0 0.0 0.0 0.0]
 [12.0 39916800.0 120541840.0 150917976.0 100259076.0 45995730.0 13339535.0 2537558.0 357423.0 32670.0 1925.0 66.0 1.0 0.0 0.0]
 [13.0 47900160.0 146644280.0 193155952.0 141401488.0 55720636.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0 1.0 0.0 0.0]
 [14.0 622702080.0 1980279040.0 2659671056.0 2031373096.0 9597703756.0 336118766.0 790943153.0 135036473.0 16669653.0 14744743.0 91091.0 3731.0 91.0 1.0 0.0]
 [15.0 8717021200.0 28345647360.0 39215679324.0 31098266400.0 159771605600.0 56663366760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 1432525.0 5005.0 105.0 1.0]]

COUNTACAS PARA SEGUINDO MENOR NUMERO
[[1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [2.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [3.0 2.0 3.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [4.0 3.0 11.0 6.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [5.0 24.0 50.0 35.0 10.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [6.0 120.0 274.0 225.0 85.0 15.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [7.0 720.0 1764.0 1624.0 735.0 175.0 21.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [8.0 5040.0 13068.0 11312.0 6769.0 1960.0 322.0 28.0 1.0 0.0 0.0 0.0 0.0 0.0]
 [9.0 40320.0 109584.0 110124.0 67284.0 22449.0 4536.0 546.0 36.0 1.0 0.0 0.0 0.0 0.0]
 [10.0 362880.0 1026576.0 1172700.0 723680.0 269325.0 62373.0 9450.0 870.0 45.0 1.0 0.0 0.0 0.0]
 [11.0 3628800.0 10622640.0 12735376.0 9409500.0 3410930.0 902085.0 157773.0 10150.0 1320.0 55.0 1.0 0.0 0.0]
 [12.0 39916800.0 120541840.0 150917976.0 100259076.0 45995730.0 13339535.0 2537558.0 357423.0 32670.0 1925.0 66.0 1.0 0.0 0.0]
 [13.0 47900160.0 146644280.0 193155952.0 141401488.0 55720636.0 206070150.0 44990231.0 6926634.0 749463.0 55770.0 2717.0 78.0 1.0 0.0 0.0]
 [14.0 622702080.0 1980279040.0 2659671056.0 2031373096.0 9597703756.0 336118766.0 790943153.0 135036473.0 16669653.0 14744743.0 91091.0 3731.0 91.0 1.0 0.0]
 [15.0 8717021200.0 28345647360.0 39215679324.0 31098266400.0 159771605600.0 56663366760.0 14409322928.0 2681453775.0 368411615.0 37312275.0 2749747.0 1432525.0 5005.0 105.0 1.0]]
```

### 6- Cálculo maior/segundo maior e menor/segundo menor utilizando *divide and conquer* e o fatorial através de *memoization*

A quarta versão do programa, utiliza *divide and conquer* para o cálculo do maior/segundo maior e menor/segundo menor para os números de 1 à 10. Observa-se que o número de trocas corresponde ao fatorial de  $n$ . Dito isto dos números de 11 a 15 é realizado o cálculo do fatorial através do memoization. Observa-se que a soma dos valores de cada linha do método anterior também corresponde ao fatorial apresentado nesta nova versão do programa.

Para a execução é utilizado em torno de 1Gb de memória, utilizado para armazenamento da lista de permutações de cada número.

Código Fonte:

[https://github.com/dasamerica/ct208/blob/master/aula\\_2805\\_maior\\_elemento/ct208\\_maior\\_men](https://github.com/dasamerica/ct208/blob/master/aula_2805_maior_elemento/ct208_maior_men)  
or segundo elemento DCM.py

Programa Principal	Funções
<pre># Programa phyton para dado um numero gera uma lista de suas permutações # Verifica quantas iterações necessárias para achar o maior, menor, segundo maior e segundo menor elemento da lista # Daniela América da Silva # Disciplina: ct208</pre>	<pre># Programa phyton para dado um numero gera uma lista de suas permutações # Verifica quantas iterações necessárias para achar o maior, menor, segundo maior e segundo menor elemento da lista # Daniela América da Silva # Disciplina: ct208  # Importa library de permutações</pre>

```

# inicializa matriz t que armazena resultados
das trocas
t_maior = np.zeros((15, 16))
t_maior[0][0] = 1

t_menor = np.zeros((15, 16))
t_menor[0][0] = 1

t_maior_segundo = np.zeros((15, 16))
t_maior_segundo[0][0] = 1

t_menor_segundo = np.zeros((15, 16))
t_menor_segundo[0][0] = 1

#processa lista de permutações do número 2 ao
10
x = 2
while x < 11:
    # inicializa lista de trocas
    trocas = []

    numeros = cria_lista (x)

    #inicializa contador trocas
    conta_trocas_maior = set_contador(15)
    conta_trocas_menor = set_contador(15)
    conta_trocas_maior_segundo =
set_contador(15)
    conta_trocas_menor_segundo =
set_contador(15)

    # realiza as permutações
    perm = permutations(numeros)

    # Conta as trocas
    for i in list(perm):
        #print (i)
        number = maior_menor_trocas (i)
        #imprime
        #print (i, '(', number ,')')
        conta_trocas_maior[number[0]]+=1
        conta_trocas_menor[number[1]]+=1
        conta_trocas_maior_segundo[number[2]]+=1
        conta_trocas_menor_segundo[number[3]]+=1

    #print (trocas)
    #print ('numero:', x, conta_trocas)

```

```

from itertools import permutations
# Importa library para matriz
import numpy as np
# Importa library para truncate
import math

#seta vetor contador de trocas
def set_contador (col):
    n = 0
    conta = []
    while n < col:
        conta.append(0)
        n+=1
    return (conta)

#cria a lista de números
def cria_lista (num):
    n = 0
    lista = []
    while n < num:
        lista.append(n+1)
        n+=1
    return (lista)

#divide and conquer min
def two_min(arr, iter):
    n = len(arr)
    m=math.trunc(n/2)

    #print(arr, n, iter)
    if n==2: # Oops, we don't consider this
as comparison, right?

        if arr[0]<arr[1]:
            #
Line 1
            iter+=1
            return (arr[0], arr[1], iter)
        else:
            iter+=1
            return (arr[1], arr[0], iter)
    if m == 1:
        if arr[0]<arr[1]:
            (least_left, sec_least_left) =
(arr[0],arr[1])
        else:
            (least_left, sec_least_left) =
(arr[1],arr[0])
    else: # always compare at least pairs

```



```

#atualiza matriz trocas
j = 0
t_maior[x-1][j]=x
t_menor[x-1][j]=x
t_maior_segundo[x-1][j]=x
t_menor_segundo[x-1][j]=x

j = 1
while j < x+1:
    t_maior[x-1][j] = conta_trocas_maior[j-1]
    t_menor[x-1][j] = conta_trocas_menor[j-1]
    t_maior_segundo[x-1][j] =
conta_trocas_maior_segundo[j-1]
    t_menor_segundo[x-1][j] =
conta_trocas_menor_segundo[j-1]
    j+=1

x+=1

#utilizando divide and conquer o número de
iterações é o fatorial do número
#é possível utilizar memoization para
calcular o fatorial a partir do número 11

#processa lista de permutações do número 11
ao 15, utilizando a recorrência
l=10
while l < 15:
    t_maior[l][0] = t_menor[l][0] =
t_maior_segundo[l][0] = t_menor_segundo[l][0]
    = l+1
    t_maior[l][l+1] = t_menor[l][l+1] =
t_maior_segundo[l][l+1] =
t_menor_segundo[l][l+1] = factorial(l+1)
    l+=1

#np.set_printoptions(precision=10,
suppress=True, linewidth=10000)

np.set_printoptions(formatter={'float':
lambda x: ' ' + str(x)},linewidth=10000 )

print ('CONTA TROCAS PARA MAIOR NUMERO')
print (t_maior)
print ('CONTA TROCAS PARA MENOR NUMERO')
print (t_menor)

```

```

    (least_left, sec_least_left, iter) =
two_min(arr[0:m], iter)
    (least_right, sec_least_right, iter) =
two_min(arr[m:n], iter)
    if least_left < least_right: #
Line 2
        iter+=1
        least = least_left
        if least_right < sec_least_left: #
Line 3
            return (least, least_right, iter)
        else:
            return (least, sec_least_left,
iter)
    else:
        iter+=1
        least = least_right
        if least_left < sec_least_right: #
Line 4
            return (least, least_left, iter)
        else:
            return (least, sec_least_right,
iter)

#divide and conquer max
def two_max(arr, iter):
    n = len(arr)
    m=math.trunc(n/2)

    #print(arr, n, iter)
    if n==2: # Oops, we don't consider this
as comparison, right?

        if arr[0]>arr[1]: #
Line 1
            iter+=1
            return (arr[0], arr[1], iter)
        else:
            iter+=1
            return (arr[1], arr[0], iter)
    if m == 1:
        if arr[0]>arr[1]:
            (biggest_left, sec_biggest_left) =
(arr[0],arr[1])
        else:
            (biggest_left, sec_biggest_left) =
(arr[1],arr[0])

```

```

print ('CONTA TROCAS PARA SEGUNDO MAIOR
NUMERO')
print (t_maior_segundo)
print ('CONTA TROCAS PARA SEGUNDO MENOR
NUMERO')
print (t_menor_segundo)

```

```

        else: # always compare at least pairs
            (biggest_left, sec_biggest_left, iter)
= two_max(arr[0:m], iter)
            (biggest_right, sec_biggest_right, iter)
= two_max(arr[m:n], iter)
            if biggest_left > biggest_right:
# Line 2
                iter+=1
                biggest = biggest_left
                if biggest_right > sec_biggest_left:
# Line 3
                    return (biggest, biggest_right,
iter)
            else:
                return (biggest,
sec_biggest_left, iter)
            else:
                iter+=1
                biggest = biggest_right
                if biggest_left > sec_biggest_right:
# Line 4
                    return (biggest, biggest_left,
iter)
            else:
                return (biggest,
sec_biggest_right, iter)

#conta numero de trocas incluindo o segundo
elemento
#utiliza divide and conquer
def maior_menor_trocas (elements):
    me = 0
    ma = 0
    ma_2 = 0
    me_2 = 0
    t_ma = 0
    t_me = 0
    t= [0,0,0,0]

    (ma, ma_2, t_ma) = two_max(elements,-1)
    t[0]=t_ma
    t[2]=t_ma

    (me, me_2, t_me) = two_min(elements,-1)
    t[1]=t_me
    t[3]=t_me

    return (t)

```



```

[ 2.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 3.0  0.0  6.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 4.0  0.0  0.0  24.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 5.0  0.0  0.0  0.0  120.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 6.0  0.0  0.0  0.0  0.0  720.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 7.0  0.0  0.0  0.0  0.0  0.0  5040.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 8.0  0.0  0.0  0.0  0.0  0.0  0.0  40320.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 9.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  362880.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 10.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  3628800.0  0.0  0.0  0.0  0.0  0.0  0.0]
[ 11.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  39916800.0  0.0  0.0  0.0  0.0  0.0]
[ 12.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  479001600.0  0.0  0.0  0.0  0.0]
[ 13.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  6227020800.0  0.0  0.0  0.0]
[ 14.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  87178291200.0  0.0  0.0]
[ 15.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1307674368000.0]]

CONTA TROCAS PARA SEGUNDO MENOR NUMERO
[[ 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 2.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 3.0  0.0  6.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 4.0  0.0  0.0  24.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 5.0  0.0  0.0  0.0  120.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 6.0  0.0  0.0  0.0  0.0  720.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 7.0  0.0  0.0  0.0  0.0  0.0  5040.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 8.0  0.0  0.0  0.0  0.0  0.0  0.0  40320.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 9.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  362880.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 10.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  3628800.0  0.0  0.0  0.0  0.0  0.0  0.0]
 [ 11.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  39916800.0  0.0  0.0  0.0  0.0  0.0]
 [ 12.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  479001600.0  0.0  0.0  0.0  0.0]
 [ 13.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  6227020800.0  0.0  0.0  0.0]
 [ 14.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  87178291200.0  0.0  0.0]
 [ 15.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1307674368000.0]]

```

**Trocas Maior e segundo maior**

```
colab.research.google.com/drive/1KqmHn1PmnDEORCs_88LspY_45TXhEtum
Biblioteca ITA - Divi... Scopus - Document... Web of Science [v.5... Softwares Bibliométr... Parsifal 1a Reunião Projeto... Prezados Senhores, VOSviewer - Visuali...
+ Code + Text RAM Disk Editing
CONTA TROCAS PARA MAIOR NUMERO
[[ 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 2.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 3.0 0.0 6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 4.0 0.0 0.0 24.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 5.0 0.0 0.0 0.0 120.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 6.0 0.0 0.0 0.0 0.0 720.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 7.0 0.0 0.0 0.0 0.0 0.0 5040.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 8.0 0.0 0.0 0.0 0.0 0.0 0.0 40320.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 9.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 362880.0 0.0 0.0 0.0 0.0 0.0 0.0]
[10.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 3628800.0 0.0 0.0 0.0 0.0 0.0]
[11.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 39916800.0 0.0 0.0 0.0 0.0]
[12.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 479001600.0 0.0 0.0 0.0]
[13.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 6227020800.0 0.0 0.0]
[14.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 87178291200.0 0.0]
[15.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1307674368000.0]]
CONTA TROCAS PARA MENOR NUMERO
[[ 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 2.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 3.0 0.0 6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 4.0 0.0 0.0 24.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 5.0 0.0 0.0 0.0 120.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 6.0 0.0 0.0 0.0 0.0 720.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 7.0 0.0 0.0 0.0 0.0 0.0 5040.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 8.0 0.0 0.0 0.0 0.0 0.0 0.0 40320.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 9.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 362880.0 0.0 0.0 0.0 0.0 0.0 0.0]
[10.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 3628800.0 0.0 0.0 0.0 0.0 0.0]
[11.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 39916800.0 0.0 0.0 0.0 0.0]
[12.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 479001600.0 0.0 0.0 0.0]
[13.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 6227020800.0 0.0 0.0]
[14.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 87178291200.0 0.0]
[15.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1307674368000.0]]
```

Trocas Menor e segundo menor

```
colab.research.google.com/drive/1KqmHn1PmnDEORCs_88LspY_45TXhEtum
Biblioteca ITA - Divi... Scopus - Document... Web of Science [v.5... Softwares Bibliométr... Parsifal 1a Reunião Projeto... Prezados Senhores, VOSviewer - Visuali...
+ Code + Text RAM Disk Editing
CONTA TROCAS PARA SEGUNDO MAIOR NUMERO
[[ 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 2.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 3.0 0.0 6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 4.0 0.0 0.0 24.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 5.0 0.0 0.0 0.0 120.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 6.0 0.0 0.0 0.0 0.0 720.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 7.0 0.0 0.0 0.0 0.0 0.0 5040.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 8.0 0.0 0.0 0.0 0.0 0.0 0.0 40320.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 9.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 362880.0 0.0 0.0 0.0 0.0 0.0 0.0]
[10.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 3628800.0 0.0 0.0 0.0 0.0 0.0]
[11.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 39916800.0 0.0 0.0 0.0 0.0]
[12.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 479001600.0 0.0 0.0 0.0]
[13.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 6227020800.0 0.0 0.0]
[14.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 87178291200.0 0.0]
[15.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1307674368000.0]]
CONTA TROCAS PARA SEGUNDO MENOR NUMERO
[[ 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 2.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 3.0 0.0 6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 4.0 0.0 0.0 24.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 5.0 0.0 0.0 0.0 120.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 6.0 0.0 0.0 0.0 0.0 720.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 7.0 0.0 0.0 0.0 0.0 0.0 5040.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 8.0 0.0 0.0 0.0 0.0 0.0 0.0 40320.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
[ 9.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 362880.0 0.0 0.0 0.0 0.0 0.0 0.0]
[10.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 3628800.0 0.0 0.0 0.0 0.0 0.0]
[11.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 39916800.0 0.0 0.0 0.0 0.0]
[12.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 479001600.0 0.0 0.0 0.0]
[13.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 6227020800.0 0.0 0.0]
[14.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 87178291200.0 0.0]
[15.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1307674368000.0]]
```

7- Notação Big O e Funções geratrizes

Será verificado em aula do dia 10-Junho

Realizado o pré-reading da aula do prof. Don Knuth e verificado as páginas de 96-104 do livro *The art of computer programming*. Vol. 3

## 8- Programas criados para testes

Para adquirir conhecimento sobre as diferentes formas de calcular Fibonacci.

Código fonte:

[https://github.com/dasamerica/ct208/blob/master/aula\\_2805\\_maior\\_elemento/ct208\\_fibonacci.py](https://github.com/dasamerica/ct208/blob/master/aula_2805_maior_elemento/ct208_fibonacci.py)

Para adquirir conhecimento sobre divide and conquer e memoization.

Código fonte:

[https://github.com/dasamerica/ct208/blob/master/aula\\_2805\\_maior\\_elemento/ct208\\_divide\\_conquer\\_memoization.py](https://github.com/dasamerica/ct208/blob/master/aula_2805_maior_elemento/ct208_divide_conquer_memoization.py)

## Referências

1. Knuth, D, Stanford, Stanford Lecture - Don Knuth: The Analysis of Algorithms (2015, recreating 1969), 2015, acessado de <https://www.youtube.com/watch?v=vkUNH9r6UCI> em 08 Junho 2020
2. Knuth, Donald Ervin. *The art of computer programming*. Vol. 3. Pearson Education, 1997, páginas de 96-104
3. Soma, N.Y., ITA, Matemática Computacional, CT208 Notas da Aula de 28 de Maio 2020 - Maior elemento, 2020
4. Sanches, C.A.A, ITA, Estruturas de Dados, Análise de Algoritmos e Complexidade Estrutural, 2020, acessado de <http://www.comp.ita.br/~alonso/ensino/CT234/CT234-Cap10.pdf> em 08 Junho 2020.
5. Tech with Tim, You tube, Recursion and Memoization Tutorial Python, 2017, acessado de <https://www.youtube.com/watch?v=sCecRPSQg6Y> em 08 junho 2020
6. StackOverflow, StackOverflow, Memoization Fibonacci Algorithm in Python, acessado de <https://stackoverflow.com/questions/29570870/memoization-fibonacci-algorithm-in-python> em 08 Junho 2020
7. StackOverflow, StackOverflow, Finding the second smallest number from the given list using divide-and-conquer, acessado de <https://stackoverflow.com/questions/19415821/finding-the-second-smallest-number-from-the-given-list-using-divide-and-conquer/19424799#19424799> em 08 Junho 2020
8. StackOverflow, StackOverflow, What is memoization and how can I use it in Python, acessado de <https://stackoverflow.com/questions/1988804/what-is-memoization-and-how-can-i-use-it-in-python> em 08 Junho 2020