

# Solução com persistência de dados

```
In [1]: def listaProximaJogada(posicaoAtual, colunaMAX):  
        if posicaoAtual == 0:  
            return [0, 1]  
        elif posicaoAtual == colunaMAX:  
            return [colunaMAX - 1, colunaMAX]  
        elif 0 < posicaoAtual < colunaMAX:  
            return [posicaoAtual - 1, posicaoAtual, posicaoAtual + 1]  
        else:  
            return [-1]
```

```
In [2]: from itertools import permutations, product
```

```
In [3]: def produtoFiltrado():  
        D = [1, 2, 3, 4, 5, 6]  
        produto = list(product(D, D, D))  
        produto_filtrado = []  
        while (produto):  
            produto_filtrado.append(produto[0])  
            permutacoes = list(permutations([produto[0][0], produto[0][1], produto[0][2]]))  
            for p in permutacoes:  
                if p in produto:  
                    produto.remove(p)  
        return produto_filtrado
```

```
In [4]: import os  
        import shelve
```

```
In [5]: path = os.path.join('.', 'variaveis')
```

```
In [6]: if not os.path.exists(path):  
        variaveis = shelve.open(path)  
        print(f'Arquivo variaveis gerado no diretório atual: {path}')  
        variaveis.close()  
else:  
    print(f'0 arquivo variaveis já existe no diretório atual: {path}')
```

0 arquivo variaveis já existe no diretório atual: ./variaveis

```
In [7]: variaveis = shelve.open(path)
```

```
In [8]: try:  
        if variaveis['configuracoes_dados']:  
            print(f'A variável configuracoes_dados já existe em {path}.')  
except KeyError:  
    variaveis['configuracoes_dados'] = produtoFiltrado()  
    print(f'Variável configuracoes_dados gravada com sucesso em {path}.')
```

A variável configuracoes\_dados já existe em ./variaveis.

```
In [9]: def resultadosExpressoes(configuracoes_dados):
    resultados = []
    for p in configuracoes_dados:
        if p[0] == p[1] and p[1] == p[2]:
            resultados.append(p[0] + p[1] + p[2])
            resultados.append(p[0] + p[1] - p[2])
            resultados.append(p[0] - p[1] - p[2])
        elif p[0] == p[1]:
            resultados.append(p[0] + p[1] + p[2])
            resultados.append(p[0] + p[1] - p[2])
            resultados.append(p[0] - p[1] + p[2])
            resultados.append(p[0] - p[1] - p[2])
            resultados.append(p[2] - p[0] - p[1])
        elif p[1] == p[2]:
            resultados.append(p[0] + p[1] + p[2])
            resultados.append(p[0] + p[1] - p[2])
            resultados.append(p[0] - p[1] - p[2])
            resultados.append(p[1] - p[0] + p[2])
            resultados.append(p[1] - p[0] - p[2])
        elif p[0] == p[2]:
            resultados.append(p[0] + p[1] + p[2])
            resultados.append(p[0] + p[1] - p[2])
            resultados.append(p[0] - p[1] + p[2])
            resultados.append(p[0] - p[1] - p[2])
            resultados.append(p[1] - p[0] - p[2])
        else:
            resultados.append(p[0] + p[1] + p[2])
            resultados.append(p[0] + p[1] - p[2])
            resultados.append(p[0] - p[1] + p[2])
            resultados.append(p[0] - p[1] - p[2])
            resultados.append(p[1] - p[0] + p[2])
            resultados.append(p[1] - p[0] - p[2])
            resultados.append(p[2] - p[0] - p[1])
    return resultados
```

```
In [10]: try:
    if variaveis['resultados_expressoes']:
        print(f'A variável resultados_expressoes já existe em {path}.')
except KeyError:
    variaveis['resultados_expressoes'] = sorted(resultadosExpressoes(variaveis['configuracoes_dados']))
    print(f'Variável resultados_expressoes gravada com sucesso em {path}.')
```

A variável resultados\_expressoes já existe em ./variaveis.

```
In [11]: def cardinalidade(evento, resultados_expressoes):
    cardinalidade = 0
    for r in resultados_expressoes:
        if r == evento:
            cardinalidade += 1
    return cardinalidade

def probabilidade(evento, resultados_expressoes):
    return cardinalidade(evento, resultados_expressoes) / len(resultados_expressoes)
```

```
In [12]: def tabProba(resultados_expressoes):
    tab = [[7, 5, 6, 9, 4, 2, 8, 1, 3],
            [2, 8, 1, 8, 10, 7, 9, 4, 5],
            [7, 3, 2, 1, 5, 4, 5, 7, 3],
            [5, 8, 7, 2, 8, 7, 6, 9, 8],
            [7, 3, 2, 1, 5, 4, 5, 7, 3],
            [2, 4, 8, 5, 9, 7, 6, 8, 5],
            [8, 7, 3, 6, 4, 1, 2, 5, 1],
            [6, 2, 5, 7, 8, 7, 6, 4, 3],
            [8, 7, 6, 3, 5, 4, 9, 2, 7],
            [5, 4, 3, 8, 9, 1, 2, 5, 4],
            [2, 9, 7, 4, 6, 8, 7, 5, 9]]
    tab_proba = [[] for _ in range(11)]
    for lin in range(len(tab)):
        for col in range(len(tab[0])):
            tab_proba[lin].append([lin, col, tab[lin][col],
                                    probabilidade(tab[lin][col],
                                                    resultados_expressoes)])
    return tab_proba
```

```
In [13]: try:
        if variaveis['tab_proba']:
            print(f'A variável tab_proba já existe em {path}.')
        except KeyError:
            variaveis['tab_proba'] = tabProba(variaveis['resultados_expressoes'])
            print(f'Variável tab_proba gravada com sucesso em {path}.')
```

A variável tab\_proba já existe em ./variaveis.

```
In [14]: def probaCaminho(caminho):
        produto = 1
        for item in caminho:
            produto *= item[3]
        return produto

        def mapaTabuleiro(tab_proba):
            colunaMAX = 8
            mapa = []
            for p1 in [0, 1, 2, 3, 4, 5, 6, 7, 8]:
                for p2 in listaProximaJogada(p1, colunaMAX):
                    for p3 in listaProximaJogada(p2, colunaMAX):
                        for p4 in listaProximaJogada(p3, colunaMAX):
                            for p5 in listaProximaJogada(p4, colunaMAX):
                                for p6 in listaProximaJogada(p5, colunaMAX):
                                    for p7 in listaProximaJogada(p6, colunaMAX):
                                        for p8 in listaProximaJogada(p7, colunaMAX):
                                            for p9 in listaProximaJogada(p8, colunaMAX):
                                                for p10 in listaProximaJogada(p9, colunaMAX):
                                                    for p11 in listaProximaJogada(p10, colunaMAX):
                                                        caminho = [
                                                            tab_proba[0][p1], tab_proba[1][p2],
                                                            tab_proba[2][p3], tab_proba[3][p4],
                                                            tab_proba[4][p5], tab_proba[5][p6],
                                                            tab_proba[6][p7], tab_proba[7][p8],
                                                            tab_proba[8][p9], tab_proba[9][p10],
                                                            tab_proba[10][p11]
                                                        ]
                                                        proba = probaCaminho(caminho)
                                                        mapa.append((caminho, proba))

            return mapa
```

```
In [15]: try:
        if variaveis['mapa_tabuleiro']:
            print(f'A variável mapa_tabuleiro já existe em {path}.')
        except KeyError:
            variaveis['mapa_tabuleiro'] = mapaTabuleiro(variaveis['tab_proba'])
            print(f'Variável mapa_tabuleiro gravada com sucesso em {path}.')
```

A variável mapa\_tabuleiro já existe em ./variaveis.

```
In [16]: def distintasProbas(mapa_tabuleiro):
        distintas_probas = []
        for caminho in mapa_tabuleiro:
            if caminho[1] not in distintas_probas:
                distintas_probas.append(caminho[1])
        return distintas_probas
```

```
In [17]: try:
        if variaveis['distintas_probas']:
            print(f'A variável distintas_probas já existe em {path}.')
        except KeyError:
            variaveis['distintas_probas'] = distintasProbas(variaveis['mapa_tabuleiro'])
            print(f'Variável distintas_probas gravada com sucesso em {path}.')
```

A variável distintas\_probas já existe em ./variaveis.

```
In [18]: def mapaFaixaProbas(mapa_tabuleiro, distintas_probas, indice_min_proba, indice_max_proba):
        distintas_probas.sort()
        inf = distintas_probas[indice_min_proba]
        sup = distintas_probas[indice_max_proba]
        caminhos = []
        for item in mapa_tabuleiro:
            if inf <= item[1] <= sup:
                caminhos.append(item)
        return caminhos
```

```
In [19]: try:
         if variaveis['submapa_inferior']:
             print(f'A variável submapa_inferior já existe em {path}.')
         except KeyError:
             variaveis['submapa_inferior'] = mapaFaixaProbas(variaveis['mapa_tabuleiro'], variaveis['dis']
             print(f'Variável submapa_inferior gravada com sucesso em {path}.')
```

A variável submapa\_inferior já existe em ./variaveis.

```
In [20]: try:
         if variaveis['submapa_superior']:
             print(f'A variável submapa_superior já existe em {path}.')
         except KeyError:
             variaveis['submapa_superior'] = mapaFaixaProbas(variaveis['mapa_tabuleiro'], variaveis['dis']
             print(f'Variável submapa_superior gravada com sucesso em {path}.')
```

A variável submapa\_superior já existe em ./variaveis.

```
In [21]: import random as rd
```

```

In [22]: def subMapaAleatorio(submapa, quantidade):
    lista = []
    if len(submapa) >= quantidade:
        for caminho in rd.sample(submapa, quantidade):
            lista.append(caminho)
    return lista

def total(submapa_inferior, submapa_superior):
    linf = len(submapa_inferior)
    lsup = len(submapa_superior)
    if linf < lsup:
        return linf
    return lsup

def mapaRazoes(submapa_inferior, submapa_superior):
    tab_razoes = [[1,1] for _ in range(9)] for _ in range(11)]
    razoes = [[1 for _ in range(9)] for _ in range(11)]

    for caminho_prob in submapa_inferior:
        caminho = caminho_prob[0]
        for casa in caminho:
            lin = casa[0]
            col = casa[1]
            tab_razoes[lin][col][1] += 1
    for caminho_prob in submapa_superior:
        caminho = caminho_prob[0]
        for casa in caminho:
            lin = casa[0]
            col = casa[1]
            tab_razoes[lin][col][0] += 1
    for l in range(11):
        for c in range(9):
            razoes[l][c] = tab_razoes[l][c][0] / tab_razoes[l][c][1]
    return razoes

def mediasRazoes(lista_mapas_razoes):
    tam = len(lista_mapas_razoes)
    medias = [[0 for _ in range(9)] for _ in range(11)]
    for mapa_razoes in lista_mapas_razoes:
        for i in range(11):
            for j in range(9):
                medias[i][j] += mapa_razoes[i][j]
    for i in range(11):
        for j in range(9):
            medias[i][j] = medias[i][j] / tam
    return medias

def desviosRazoes(lista_mapas_razoes, medias):
    tam = len(lista_mapas_razoes)
    desvios = [[0 for _ in range(9)] for _ in range(11)]
    for mapa_razoes in lista_mapas_razoes:
        for i in range(11):
            for j in range(9):
                desvios[i][j] += abs(mapa_razoes[i][j] - medias[i][j])
    for i in range(11):
        for j in range(9):
            desvios[i][j] = desvios[i][j] / tam
    return desvios

def mediasComDesvio(medias, desvios, sup):
    medias_desvio = [[0 for _ in range(9)] for _ in range(11)]
    if sup:
        for i in range(11):
            for j in range(9):
                medias_desvio[i][j] = medias[i][j] + desvios[i][j]
    else:
        for i in range(11):
            for j in range(9):
                medias_desvio[i][j] = medias[i][j] - desvios[i][j]
    return medias_desvio

```

```
In [23]: def calculaMapas(TOTAL):
        lista_mapas_razoes = []
        for _ in range(2000):
            submapa_aleatorio_superior = subMapaAleatorio(variaveis['submapa_superior'], TOTAL)
            submapa_aleatorio_inferior = subMapaAleatorio(variaveis['submapa_inferior'], TOTAL)
            razoes = mapaRazoes(submapa_aleatorio_inferior, submapa_aleatorio_superior)
            lista_mapas_razoes.append(razoes)
        return lista_mapas_razoes
```

```
In [24]: try:
        if variaveis['total']:
            print(f'A variável total já existe em {path}.')
    except KeyError:
        variaveis['total'] = total(variaveis['submapa_inferior'], variaveis['submapa_superior'])
        print(f'Variável total gravada com sucesso em {path}.')
```

A variável total já existe em ./variaveis.

```
In [25]: try:
        if variaveis['lista_mapas_razoes']:
            print(f'A variável lista_mapas_razoes já existe em {path}.')
    except KeyError:
        variaveis['lista_mapas_razoes'] = calculaMapas(variaveis['total'])
        print(f'Variável lista_mapas_razoes gravada com sucesso em {path}.')
```

A variável lista\_mapas\_razoes já existe em ./variaveis.

```
In [26]: try:
        if variaveis['medias']:
            print(f'A variável medias já existe em {path}.')
    except KeyError:
        variaveis['medias'] = mediasRazoes(variaveis['lista_mapas_razoes'])
        print(f'Variável medias gravada com sucesso em {path}.')
```

A variável medias já existe em ./variaveis.

```
In [27]: try:
        if variaveis['desvios']:
            print(f'A variável desvios já existe em {path}.')
    except KeyError:
        variaveis['desvios'] = desviosRazoes(variaveis['lista_mapas_razoes'], variaveis['medias'])
        print(f'Variável desvios gravada com sucesso em {path}.')
```

A variável desvios já existe em ./variaveis.

```
In [28]: try:
        if variaveis['medias_inf']:
            print(f'A variável medias_inf já existe em {path}.')
    except KeyError:
        variaveis['medias_inf'] = mediasComDesvio(variaveis['medias'], variaveis['desvios'], False)
        print(f'Variável medias_inf gravada com sucesso em {path}.')
```

A variável medias\_inf já existe em ./variaveis.

```
In [29]: try:
        if variaveis['medias_sup']:
            print(f'A variável medias_sup já existe em {path}.')
    except KeyError:
        variaveis['medias_sup'] = mediasComDesvio(variaveis['medias'], variaveis['desvios'], True)
        print(f'Variável medias_sup gravada com sucesso em {path}.')
```

A variável medias\_sup já existe em ./variaveis.

```
In [30]: from matplotlib import pyplot as plt
```

```

In [31]: fig = plt.figure()
ax1 = fig.add_subplot(1, 3, 1)
ax2 = fig.add_subplot(1, 3, 2)
ax3 = fig.add_subplot(1, 3, 3)

VMAX = 10

im = ax1.pcolor(variaveis['medias_inf'], cmap="viridis_r", vmin=0, vmax=VMAX)
ax1.set_title('Médias Inferiores\n', fontsize=18)

ax2.pcolor(variaveis['medias'], cmap="viridis_r", vmin=0, vmax=VMAX)
ax2.set_title('Médias\n', fontsize=18)

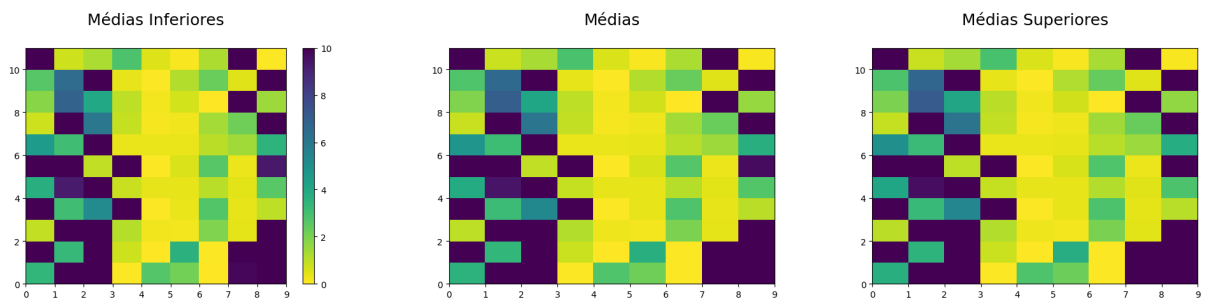
ax3.pcolor(variaveis['medias_sup'], cmap="viridis_r", vmin=0, vmax=VMAX)
ax3.set_title('Médias Superiores\n', fontsize=18)

plt.colorbar(im)

plt.subplots_adjust(right=3, wspace=0.3)

plt.show()

```



```

In [32]: def jogar(tabuleiro, configuracoes_dados, imprimir=False):
    passou = 0
    coluna = rd.randint(0, 2)
    pos_inicial = (0, coluna)

    i = 0
    while i < len(tabuleiro) - 1:
        jogada = rd.sample(configuracoes_dados, 1)
        expressoes = resultadosExpressoes(jogada)
        proxima_jogada = listaProximaJogada(pos_inicial[1], 2)
        possibilidades = []
        for posicao in proxima_jogada:
            for expressao in expressoes:
                if expressao == tabuleiro[i + 1][posicao]:
                    possibilidades.append(expressao)

        if imprimir:
            print(f'Posição inicial: {pos_inicial}')
            print(f'Jogada: {jogada}')
            print(f'Expressões: {expressoes}')
            print(f'Lista da próxima jogada: {proxima_jogada}')
            print(f'Próxima linha: {tabuleiro[i + 1]}')
            print(f'Possibilidades: {possibilidades}')

        if possibilidades:
            escolha = rd.choice(possibilidades)
            coluna = tabuleiro[i + 1].index(escolha)
            i += 1
            pos_inicial = (i, coluna)
            if imprimir:
                print(f'Escolha: {escolha}')
        else:
            passou += 1
        if imprimir:
            print(f'Jogadas sem avançar: {passou}\n')
    return passou

```

```
In [33]: tab_esquerda = [[7,5,6],[2, 8,1],[7,3,2],[5,8,7],[7,3,2],[2,4,8],[8,7,3],[6,2,5],[8,7,6],
                        [5,4,3],[2,9,7]]
tab_centro = [[9,4,2],[8,10,7],[1,5,4],[2,8,7],[1,5,4],[5,9,7],[6,4,1],[7,8,7],[3,5,4],
              [8,9,1],[4,6,8]]
tab_direita = [[8,1,3],[9, 4,5],[5,7,3],[6,9,8],[5,7,3],[6,8,5],[2,5,1],[6,4,3],[9,2,7],
               [2,5,4],[7,5,9]]
```

```
In [34]: configuracoes_dados = variaveis['configuracoes_dados']
```

```
In [35]: def simulacao(numero_testes):
    esquerda = []
    centro = []
    direita = []
    for _ in range(numero_testes):
        esquerda.append(jogar(tab_esquerda, configuracoes_dados))
        centro.append(jogar(tab_centro, configuracoes_dados))
        direita.append(jogar(tab_direita, configuracoes_dados))
    esquerda = sum(esquerda) / numero_testes
    centro = sum(centro) / numero_testes
    direita = sum(direita) / numero_testes
    return esquerda, centro, direita
```

```
In [36]: import numpy as np
import pandas as pd
from multiprocessing import Pool

pool = Pool(8)

valores = [2*10**4 for _ in range(8)]
```

```
In [37]: dados = pool.map(simulacao, valores)
```

```
In [38]: dados = np.array(dados)

df = pd.DataFrame(dados,
                  index=['2 milhoes', '4 milhoes', '6 milhoes', '8 milhoes',
                        '10 milhoes', '12 milhoes', '14 milhoes', '16 milhoes'],
                  columns=['Esquerda', 'Centro', 'Direita'])
df
```

```
Out[38]:
```

	Esquerda	Centro	Direita
2 milhoes	9.04615	11.14075	9.75580
4 milhoes	9.01350	11.20340	9.67750
6 milhoes	9.03620	11.16110	9.70215
8 milhoes	9.02030	11.25210	9.62560
10 milhoes	9.02445	11.13235	9.64930
12 milhoes	9.01730	11.19235	9.72330
14 milhoes	9.04975	11.22370	9.62780
16 milhoes	8.98800	11.12855	9.76015

```
In [39]: df.mean()
```

```
Out[39]: Esquerda      9.024456
Centro      11.179287
Direita      9.690200
dtype: float64
```

```
In [40]: df['% Centro'] = 100 * (df['Centro'] - df['Esquerda']) / df['Esquerda']
df['% Direita'] = 100 * (df['Direita'] - df['Esquerda']) / df['Esquerda']

percentual = df.drop(['Esquerda', 'Centro', 'Direita'], axis=1)

percentual
```



Out[40]:

	% Centro	% Direita
2 milhoes	23.154602	7.844774
4 milhoes	24.295779	7.366728
6 milhoes	23.515416	7.369801
8 milhoes	24.741971	6.710420
10 milhoes	23.357656	6.923968
12 milhoes	24.120857	7.829395
14 milhoes	24.022211	6.387469
16 milhoes	23.815643	8.590899

In [41]: `percentual.mean()`

Out[41]: % Centro      23.878017  
% Direita      7.377932  
dtype: float64

In [42]: `try:`  
    `if variaveis:`  
        `variaveis.close()`  
        `print(f'0 arquivo variaveis foi finalizado.')`  
`except NameError:`  
    `print(f'0 arquivo variaveis já foi finalizado.')`  
  
0 arquivo variaveis foi finalizado.