**Insertion sort (Python)**

import random

import matplotlib.pyplot as plt

import time

def insertSort(arr):

    for i in range(1, len(arr)):

        key = arr[i]

        j = i-1

        while j >= 0 and key < arr[j]:

            arr[j+1] = arr[j]

            j = j-1

        arr[j+1] = key

# Generate random arrays of integers

sizes = [1000, 10000, 100000]

times = []

for size in sizes:

    data = [random.randint(0, 100) for \_ in range(size)]

    # Sort the array and measure the elapsed time

    start\_time = time.time()

    insertSort(data)

    end\_time = time.time()

    # Append the elapsed time to the list of times

    times.append(end\_time - start\_time)

    print("Tempo decorrido no tamanho",size,":", round(end\_time - start\_time, 3), "segundos")

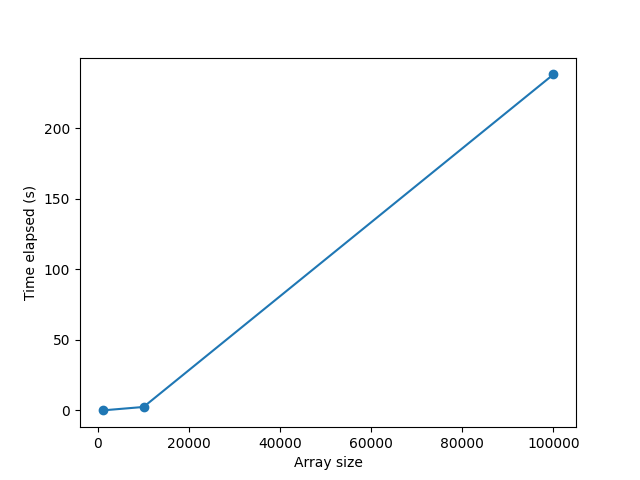
# Plot the graph

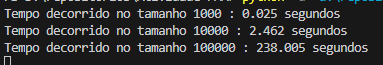
plt.plot(sizes, times, 'o-')

plt.xlabel('Array size')

plt.ylabel('Time elapsed (s)')

plt.show()





**BubbleSort (Python)**

import random

import matplotlib.pyplot as plt

import time

def bubbleSort(arr):

    for i in range(len(arr)):

        for j in range(len(arr) - 1):

            if arr[j] > arr[j+1]:

                arr[j], arr[j+1] = arr[j+1], arr[j]

# Generate random arrays of integers

sizes = [1000, 10000, 100000]

times = []

for size in sizes:

    data = [random.randint(0, 100) for \_ in range(size)]

    # Sort the array and measure the elapsed time

    start\_time = time.time()

    bubbleSort(data)

    end\_time = time.time()

    # Append the elapsed time to the list of times

    times.append(end\_time - start\_time)

    print("Tempo decorrido no tamanho",size,":", round(end\_time - start\_time, 3), "segundos")

# Plot the graph

plt.plot(sizes, times, 'o-')

plt.xlabel('Array size')

plt.ylabel('Time elapsed (s)')

plt.show()

QuickSort (Python)

import random

import time

import matplotlib.pyplot as plt

# function to partition the array on the basis of pivot element

def partition(array, low, high):

    i = (low - 1)

    pivot = array[high]

    for j in range(low, high):

        if array[j] <= pivot:

            i = i + 1

            # Swapping element at i with element at j

            (array[i], array[j]) = (array[j], array[i])

    # Swap the pivot element with the greater element specified by i

    (array[i + 1], array[high]) = (array[high], array[i + 1])

    # Return the position from where partition is done

    return i + 1

# function to perform quicksort

def quickSort(array):

    # Create an empty stack

    stack = []

    # Push initial values of low and high to stack

    stack.append((0, len(array) - 1))

    # Loop until stack is empty

    while stack:

        # Pop low and high from stack

        low, high = stack.pop()

        # Set pivot element at its correct position in sorted array

        pi = partition(array, low, high)

        # If there are elements on left side of pivot,

        # then push left side to stack

        if pi - 1 > low:

            stack.append((low, pi - 1))

        # If there are elements on right side of pivot,

        # then push right side to stack

        if pi + 1 < high:

            stack.append((pi + 1, high))

# Generate random arrays of integers

sizes = [1000, 10000, 100000]

times = []

for size in sizes:

    data = [random.randint(0, 100) for \_ in range(size)]

    # Sort the array and measure the elapsed time

    start\_time = time.time()

    quickSort(data)

    end\_time = time.time()

    # Append the elapsed time to the list of times

    times.append(end\_time - start\_time)

    print("Tempo decorrido no tamanho",size,":", round(end\_time - start\_time, 3), "segundos")

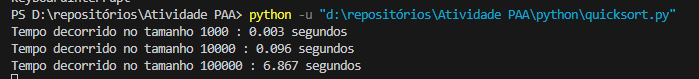
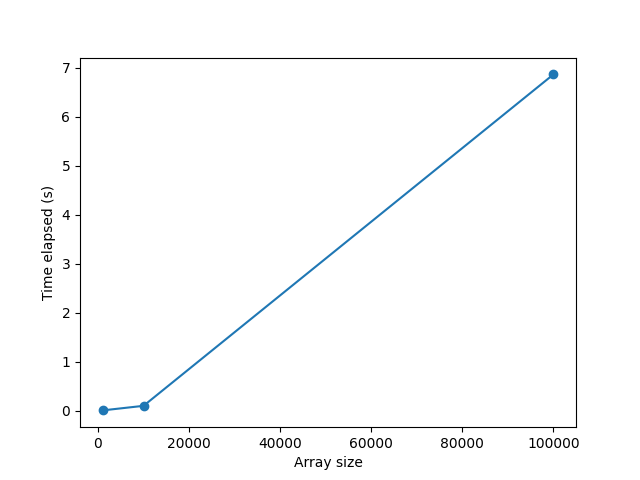
# Plot the graph

plt.plot(sizes, times, 'o-')

plt.xlabel('Array size')

plt.ylabel('Time elapsed (s)')

plt.show()



**Busca Sequencial (Python)**

import random

import re

import matplotlib.pyplot as plt

import time

def busca\_sequencial(lista, item):

    for i in range(len(lista)):

        if lista[i] == item:

            return i

    return None

# Generate random arrays of integers

sizes = [1000, 10000, 100000]

times = []

for size in sizes:

    data = [random.randint(0, 100) for \_ in range(size)]

    # Sort the array and measure the elapsed time

    start\_time = time.time()

    retorno = busca\_sequencial(data, 50)

    end\_time = time.time()

    print(retorno)

    # Append the elapsed time to the list of times

    times.append(end\_time - start\_time)

    print("Tempo decorrido no tamanho",size,":", round(end\_time - start\_time, 3), "segundos")

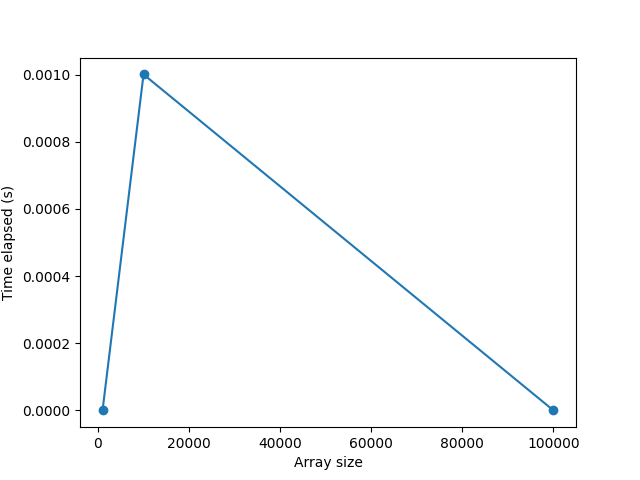
# Plot the graph

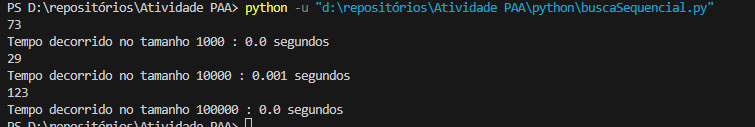
plt.plot(sizes, times, 'o-')

plt.xlabel('Array size')

plt.ylabel('Time elapsed (s)')

plt.show()





Busca binária Não recursiva (Python)

import random

import time

import matplotlib.pyplot as plt

def pesquisaBinariaNaoRecursiva(lista, item):

    lista\_ordenada = sorted(lista)

    primeiro = 0

    ultimo = len(lista\_ordenada) - 1

    while primeiro <= ultimo:

        meio = (primeiro + ultimo) // 2

        if lista\_ordenada[meio] == item:

            return meio

        else:

            if item < lista\_ordenada[meio]:

                ultimo = meio - 1

            else:

                primeiro = meio + 1

    return None

# Generate random arrays of integers

sizes = [1000, 10000, 100000]

times = []

for size in sizes:

    data = [random.randint(0, 100) for \_ in range(size)]

    # Sort the array and measure the elapsed time

    start\_time = time.time()

    retorno = pesquisaBinariaNaoRecursiva(data, 50)

    end\_time = time.time()

    print(retorno)

    # Append the elapsed time to the list of times

    times.append(end\_time - start\_time)

    print("Tempo decorrido no tamanho",size,":", round(end\_time - start\_time, 3), "segundos")

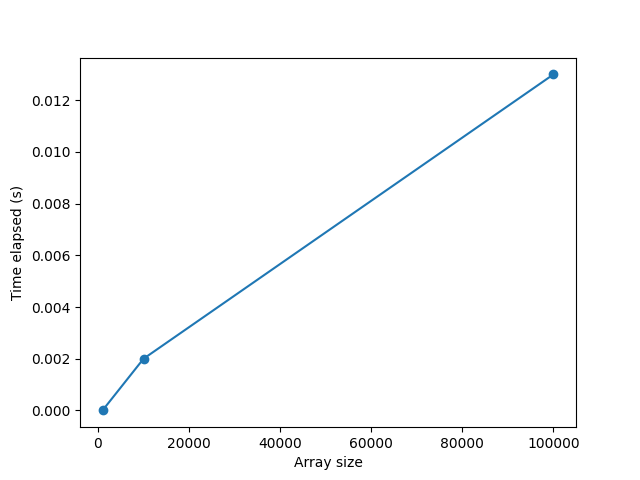
# Plot the graph

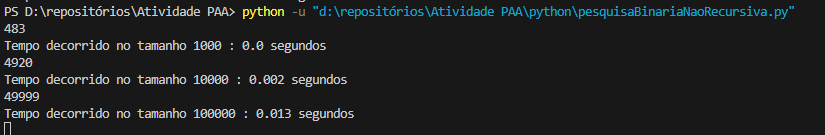
plt.plot(sizes, times, 'o-')

plt.xlabel('Array size')

plt.ylabel('Time elapsed (s)')

plt.show()





Busca Binária Recursiva (Python)

import random

import time

import matplotlib.pyplot as plt

def pesquisaBinariaRecursiva(lista, item):

    lista\_ordenada = sorted(lista)

    if len(lista\_ordenada) == 0:

        return False

    else:

        meio = len(lista\_ordenada) // 2

        if lista\_ordenada[meio] == item:

            return True

        else:

            if item < lista\_ordenada[meio]:

                return pesquisaBinariaRecursiva(lista\_ordenada[:meio], item)

            else:

                return pesquisaBinariaRecursiva(lista\_ordenada[meio + 1:], item)

# Generate random arrays of integers

sizes = [1000, 10000, 100000]

times = []

for size in sizes:

    data = [random.randint(0, 100) for \_ in range(size)]

    # Sort the array and measure the elapsed time

    start\_time = time.time()

    retorno = pesquisaBinariaRecursiva(data, 50)

    end\_time = time.time()

    print(retorno)

    # Append the elapsed time to the list of times

    times.append(end\_time - start\_time)

    print("Tempo decorrido no tamanho",size,":", round(end\_time - start\_time, 3), "segundos")

# Plot the graph

plt.plot(sizes, times, 'o-')

plt.xlabel('Array size')

plt.ylabel('Time elapsed (s)')

plt.show()

