import time

import random

# Sorting Algorithms

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

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

def selection\_sort(arr):

n = len(arr)

for i in range(n):

min\_idx = i

for j in range(i+1, n):

if arr[j] < arr[min\_idx]:

min\_idx = j

arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

def merge\_sort(arr):

if len(arr) > 1:

mid = len(arr) // 2

left = arr[:mid]

right = arr[mid:]

merge\_sort(left)

merge\_sort(right)

i = j = k = 0

while i < len(left) and j < len(right):

if left[i] < right[j]:

arr[k] = left[i]

i += 1

else:

arr[k] = right[j]

j += 1

k += 1

while i < len(left):

arr[k] = left[i]

i += 1

k += 1

while j < len(right):

arr[k] = right[j]

j += 1

k += 1

def quick\_sort(arr):

if len(arr) <= 1:

return arr

pivot = arr[len(arr) // 2]

left = [x for x in arr if x < pivot]

middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

return quick\_sort(left) + middle + quick\_sort(right)

# Function to measure execution time

def measure\_time(sort\_func, array):

start\_time = time.time()

sort\_func(array)

return time.time() - start\_time

# Array configurations

size = 1000 # size of each array

# Array 1: Best Case (already sorted)

array1 = list(range(size))

# Array 2: Average Case (random order)

array2 = random.sample(range(size), size)

# Array 3: Worst Case (reverse sorted)

array3 = list(range(size, 0, -1))

# Perform sorting and measure time

results = {

'Bubble Sort': [],

'Selection Sort': [],

'Merge Sort': [],

'Quick Sort': []

}

for sort\_name, sort\_func in zip(results.keys(), [bubble\_sort, selection\_sort, merge\_sort, quick\_sort]):

print(f"\n{sort\_name} Performance:")

for case\_name, array in zip(["Best Case", "Average Case", "Worst Case"], [array1, array2, array3]):

arr\_copy = array.copy() # copy of the array to sort

exec\_time = measure\_time(sort\_func, arr\_copy)

results[sort\_name].append(exec\_time)

print(f"{case\_name}: {exec\_time:.5f} seconds")