import random

def quicksort\_deterministic(arr, low, high):

if low < high:

pi = partition\_deterministic(arr, low, high)

quicksort\_deterministic(arr, low, pi - 1)

quicksort\_deterministic(arr, pi + 1, high)

def partition\_deterministic(arr, low, high):

pivot = arr[high]

i = low - 1

for j in range(low, high):

if arr[j] < pivot:

i += 1

arr[i], arr[j] = arr[j], arr[i]

arr[i + 1], arr[high] = arr[high], arr[i + 1]

return i + 1

def quicksort\_randomized(arr, low, high):

if low < high:

pi = partition\_randomized(arr, low, high)

quicksort\_randomized(arr, low, pi - 1)

quicksort\_randomized(arr, pi + 1, high)

def partition\_randomized(arr, low, high):

rand\_index = random.randint(low, high)

arr[rand\_index], arr[high] = arr[high], arr[rand\_index]

return partition\_deterministic(arr, low, high)

arr = list(map(int, input("Enter the elements of the array separated by space: ").split()))

print("Original Array:", arr)

arr\_deterministic = arr.copy()

quicksort\_deterministic(arr\_deterministic, 0, len(arr\_deterministic) - 1)

print("Sorted Array (Deterministic Quick Sort):", arr\_deterministic)

arr\_randomized = arr.copy()

quicksort\_randomized(arr\_randomized, 0, len(arr\_randomized) - 1)

print("Sorted Array (Randomized Quick Sort):", arr\_randomized)

import random

import time

def deterministic\_partition(low, high, a):

pivot = a[high]

i = low - 1

comparisons = 0

for j in range(low, high):

comparisons += 1

if a[j] < pivot:

i += 1

a[i], a[j] = a[j], a[i]

i += 1

a[i], a[high] = a[high], a[i]

return i, comparisons

def deterministic\_quicksort(low, high, a):

comparisons = 0

if low < high:

m, comp = deterministic\_partition(low, high, a)

comparisons += comp

comparisons += deterministic\_quicksort(low, m - 1, a)

comparisons += deterministic\_quicksort(m + 1, high, a)

return comparisons

def randomized\_partition(low, high, a):

pivot\_index = random.randint(low, high)

a[pivot\_index], a[high] = a[high], a[pivot\_index]

return deterministic\_partition(low, high, a)

def randomized\_quicksort(low, high, a):

comparisons = 0

if low < high:

m, comp = randomized\_partition(low, high, a)

comparisons += comp

comparisons += randomized\_quicksort(low, m - 1, a)

comparisons += randomized\_quicksort(m + 1, high, a)

return comparisons

if \_\_name\_\_ == "\_\_main\_\_":

a = list(map(int, input("Enter elements of the array separated by spaces: ").split()))

det\_a = a.copy()

start\_time = time.time()

det\_comparisons = deterministic\_quicksort(0, len(det\_a) - 1, det\_a)

det\_time = time.time() - start\_time

rand\_a = a.copy()

start\_time = time.time()

rand\_comparisons = randomized\_quicksort(0, len(rand\_a) - 1, rand\_a)

rand\_time = time.time() - start\_time

print("\nDeterministic Quick Sort Results:")

print("Sorted array:", det\_a)

print("Comparisons:", det\_comparisons)

print("Execution time:", det\_time)

print("\nRandomized Quick Sort Results:")

print("Sorted array:", rand\_a)

print("Comparisons:", rand\_comparisons)

print("Execution time:",rand\_time)