MOSES WAIHENYA KIRANGO

ICT-G-4-1509-21

BSCS 304 : DESIGN AND ANALYSIS OF ALGORITHMS

You are required to implement the bubble sort and merge sort algorithms using Java and then feed them with five unsorted arrays of 20, 40, 80, 160, and 320 integers, and then plot a graph to show their runtime behavior in terms of time taken in nanoseconds versus input size. You will need to also implement a random numbers generator to generate these numbers, after which you feed the numbers to the implemented algorithms. Finally, include a procedure in the code to measure the time taken in nanoseconds. After you are done, discuss the results of your experiment by comparing how the two algorithms differed as the inputs increased

```
import random

import time


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 merge_sort(arr):

    if len(arr) > 1:

        mid = len(arr) // 2

        L = arr[:mid]

        R = arr[mid:]

        merge_sort(L)

        merge_sort(R)

        i = j = k = 0

        while i < len(L) and j < len(R):

            if L[i] < R[j]:
```

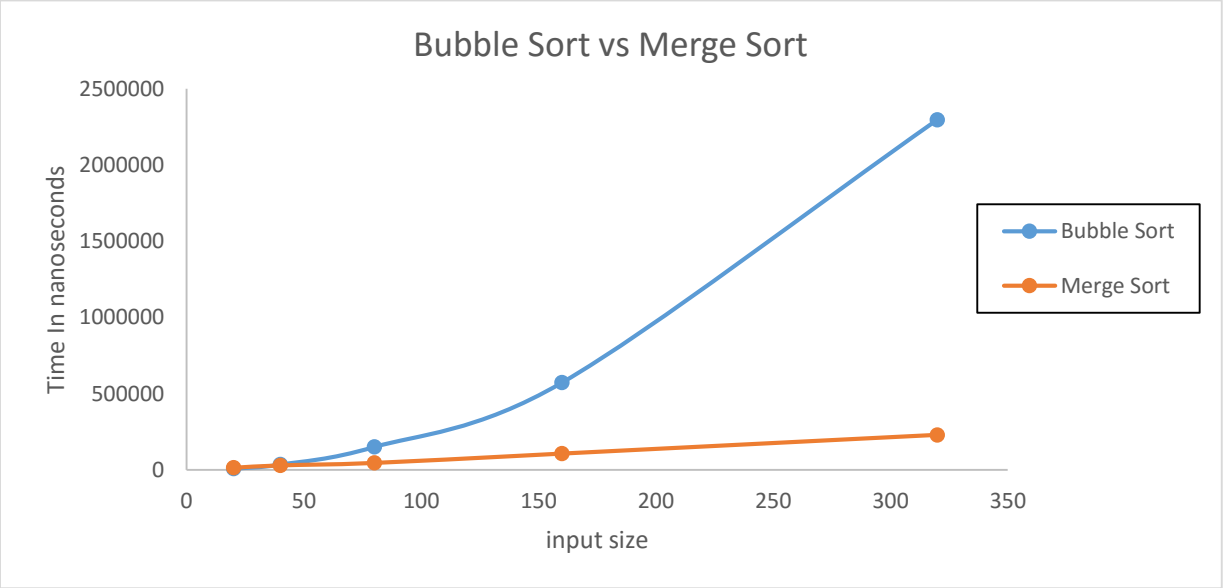
```
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1
    while i < len(L):
        arr[k] = L[i]
        i += 1
        k += 1
    while j < len(R):
        arr[k] = R[j]
        j += 1
        k += 1
def generate_arrays():
    arrays = {}
    for size in [20, 40, 80, 160, 320]:
        arrays[size] = [random.randint(1, 100) for _ in range(size)]
    return arrays

def measure_time(sort_func, arr):
    start_time = time.time_ns()
    sort_func(arr)
    end_time = time.time_ns()
    return end_time - start_time

def measure_runtime_behavior():
    arrays = generate_arrays()
    bubble_sort_times = []
```

```
    merge_sort_times = []
    for size in [20, 40, 80, 160, 320]:
        arr = arrays[size].copy()
        bubble_sort_times.append(measure_time(bubble_sort, arr))
        arr = arrays[size].copy()
        merge_sort_times.append(measure_time(merge_sort, arr))
    return bubble_sort_times, merge_sort_times
```

A GRAPH SHOWING THE RELATIONSHIP BETWEEN THE RUN TIME OF BUBBLE SORT AND MARGE SORT AGAINST INPUT SIZE



The merge sort is nearly horizontal, indicating that time taken by the algorithm is not significantly affected by the input size.

The bubble sort line is steeper, indicating that the time taken by algorithm increases more rapidly with an increase in input size.

As input size increases the time taken by bubble sort to sort the array increases exponentially while the time taken by merge sort increases at a much slower rate. This demonstrate the superiority of merge sort over bubble sort in terms of runtime complexity.