KARIYA RAJ 21162101011

# ALGORITHM ANALYSIS AND DESIGN

## PRACTICAL -5

McCormick &amp; Company is an American food company that manufactures, markets, and distributes spices, seasoning mixes, condiments, and other flavoring products for the industrial, restaurant, institutional, and home markets, they are having some number quantity of different categories item food, kindly help them to sort data using any three sorting methods and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the comparison between them. Design the algorithm for the same and implement using the programming language of your choice. Make comparative analysis for various use cases input size.

***CODE:***

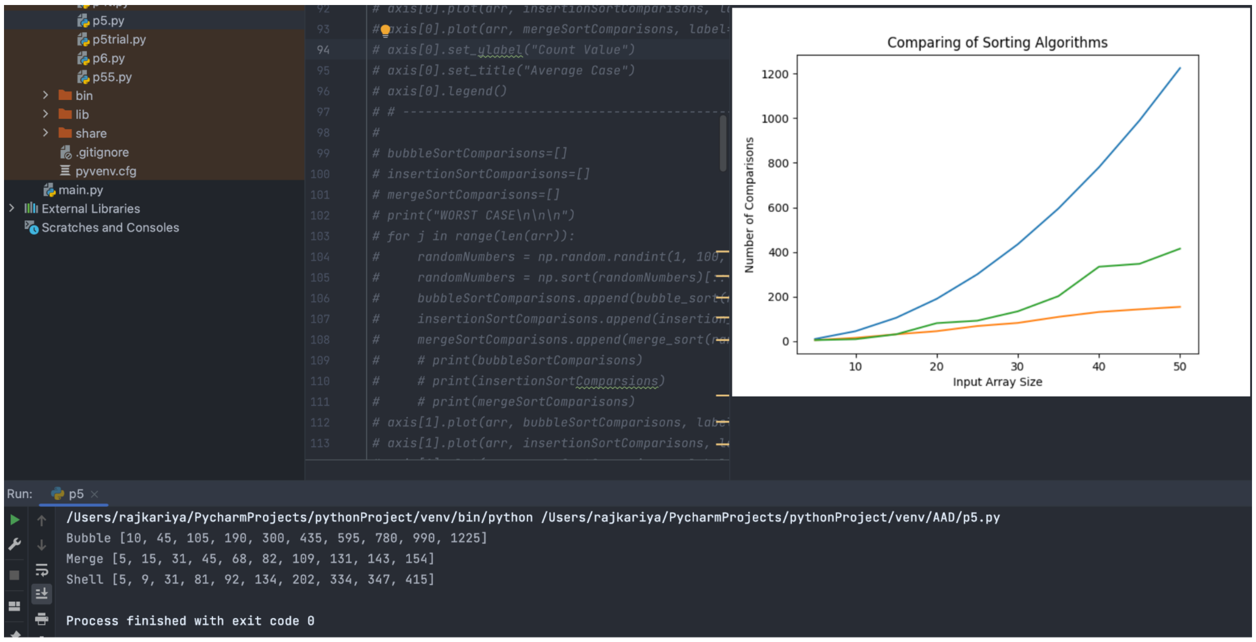
```
import random
import matplotlib.pyplot as plt
import numpy as np


def bubble_sort(arr):
n = len(arr)
comparisons = 0
for c in range(n):
for d in range(0, n - c - 1):
comparisons += 1
if arr[d] > arr[d + 1]:
arr[d], arr[d + 1] = arr[d + 1], arr[d]
return comparisons


def shell_sort(arr):
cmp = 0
n = len(arr)
gap = 1
```

```
while gap < n // 3:
gap = 3 * gap + 1

while gap >= 1:
for i in range(gap, n):
temp = arr[i]
j = i
while j >= gap and arr[j - gap] > temp:
cmp += 1
arr[j] = arr[i - gap]
j -= gap
arr[j] = temp
gap //= 3
return cmp

def merge_sort(arr):
comparisons = 0
if len(arr) > 1:
mid = len(arr) // 2
left_half = arr[:mid]
right_half = arr[mid:]

comparisons += merge_sort(left_half)
comparisons += merge_sort(right_half)

i = j = k = 0

while i < len(left_half) and j < len(right_half):
comparisons += 1
if left_half[i] < right_half[j]:
arr[k] = left_half[i]
i += 1
else:
arr[k] = right_half[j]
j += 1
k += 1

while i < len(left_half):
arr[k] = left_half[i]
i += 1
k += 1

while j < len(right_half):
arr[k] = right_half[j]
j += 1
k += 1

return comparisons
```

KARIYA RAJ 21162101011

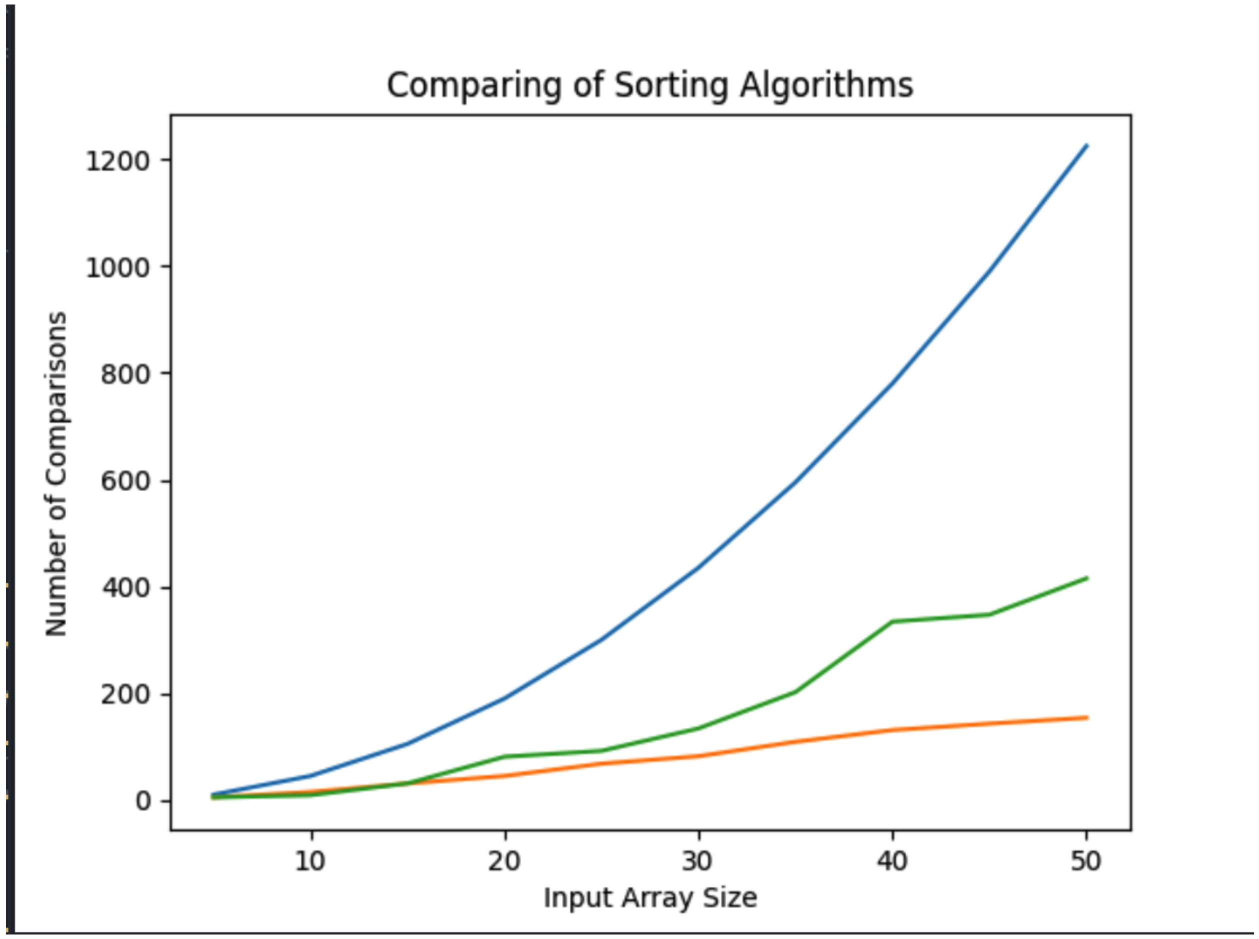
```python
arr = [5,10,15,20,25,30,35,40,45,50]
bubble = []
merge = []
shell = []
for j in range(len(arr)):
nums = np.random.randint(1, 200, arr[j])
shell.append(shell_sort(nums))
merge.append(merge_sort(nums))
bubble.append(bubble_sort(nums))


print("Bubble", bubble)
print("Merge", merge)
print("Shell", shell)
plt.plot(arr, bubble)
plt.plot(arr, merge)
plt.plot(arr, shell)
plt.ylabel("Number of Comparisons")
plt.xlabel("Input Array Size")
plt.title("Comparing of Sorting Algorithms")
plt.show()
```

***OUTPUT:***



```python
# axis[0].plot(arr, insertionSortComparisons, l
#axis[0].plot(arr, mergeSortComparisons, label=
# axis[0].set_ylabel("Count Value")
# axis[0].set_title("Average Case")
# axis[0].legend()
# # -----------------------------------------
#
# bubbleSortComparisons=[]
# insertionSortComparisons=[]
# mergeSortComparisons=[]
# print("WORST CASE\n\n\n")
# for j in range(len(arr)):
#     randomNumbers = np.random.randint(1, 100,
#     randomNumbers = np.sort(randomNumbers)[:
#     bubbleSortComparisons.append(bubble_sort(
#     insertionSortComparisons.append(insertion
#     mergeSortComparisons.append(merge_sort(ra
#     # print(bubbleSortComparisons)
#     # print(insertionSortComparsions)
#     # print(mergeSortComparisons)
# axis[1].plot(arr, bubbleSortComparisons, labe
# axis[1].plot(arr, insertionSortComparisons, l
```

```
/Users/rajkariya/PycharmProjects/pythonProject/venv/bin/python /Users/rajkariya/PycharmProjects/pythonProject/venv/AAD/p5.py
Bubble [10, 45, 105, 190, 300, 435, 595, 780, 990, 1225]
Merge [5, 15, 31, 45, 68, 82, 109, 131, 143, 154]
Shell [5, 9, 31, 81, 92, 134, 202, 334, 347, 415]

Process finished with exit code 0
```



***Questions:***

1. ***What is the best, average and worst analysis of the algorithm?***

***Ans:*** *Best Case Analysis: The best-case analysis of an algorithm refers to the scenario where the algorithm performs optimally and achieves its lowest possible time complexity. In this case, the algorithm encounters the most favorable input, and its performance is at its peak.*

*Average Case Analysis: The average-case analysis of an algorithm considers the average time complexity of the algorithm over all possible inputs. It takes into account the probability distribution of different inputs and calculates the expected performance of the algorithm.*

*Worst Case Analysis: The worst-case analysis of an algorithm examines the scenario where the algorithm performs the least efficiently and* *encounters its most unfavorable input. It provides an upper bound on the algorithm's time complexity and ensures that the algorithm will not take longer than this worst-case scenario.*

*In summary, the best-case analysis describes the most favorable performance, the average-case analysis considers the expected performance over all possible inputs, and the worst-case analysis provides the upper bound on the algorithm's performance under the least favorable circumstances.*

2. ***Which are different asymptotic notations? What is their use?***

***Ans:*** *Asymptotic notations are used to analyze algorithm efficiency based on input size.*
*- Big O (O) represents the upper bound (worst-case).*
*- Omega (Ω) represents the lower bound (best-case).*
*- Theta (Θ) provides a tight bound.*
*They aid in algorithm comparison and design, predicting scalability and simplifying complex expressions.*

3. ***What is the time Complexity of the above 3 algorithms in all cases.***

***Ans: Bubble Sort: O(n^2)***
***Merge Sort O(n logn)***
***Shell Sort O(n logn )***