Assignment 5

Shreyas Santosh More

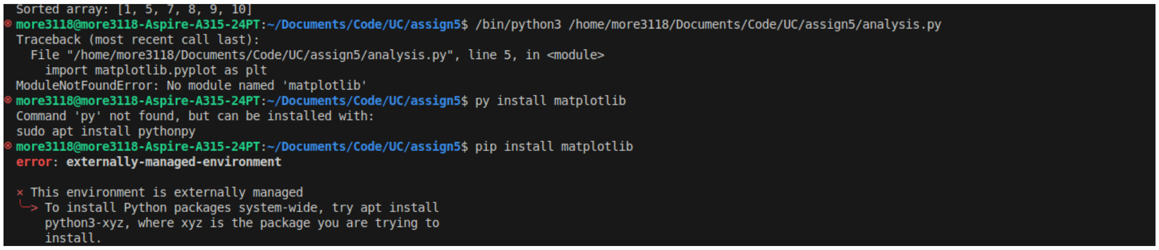
University of Cumberlands

Algorithms and Data Structures (MSCS-532-B01)

Dr.Bass

November 16, 2025

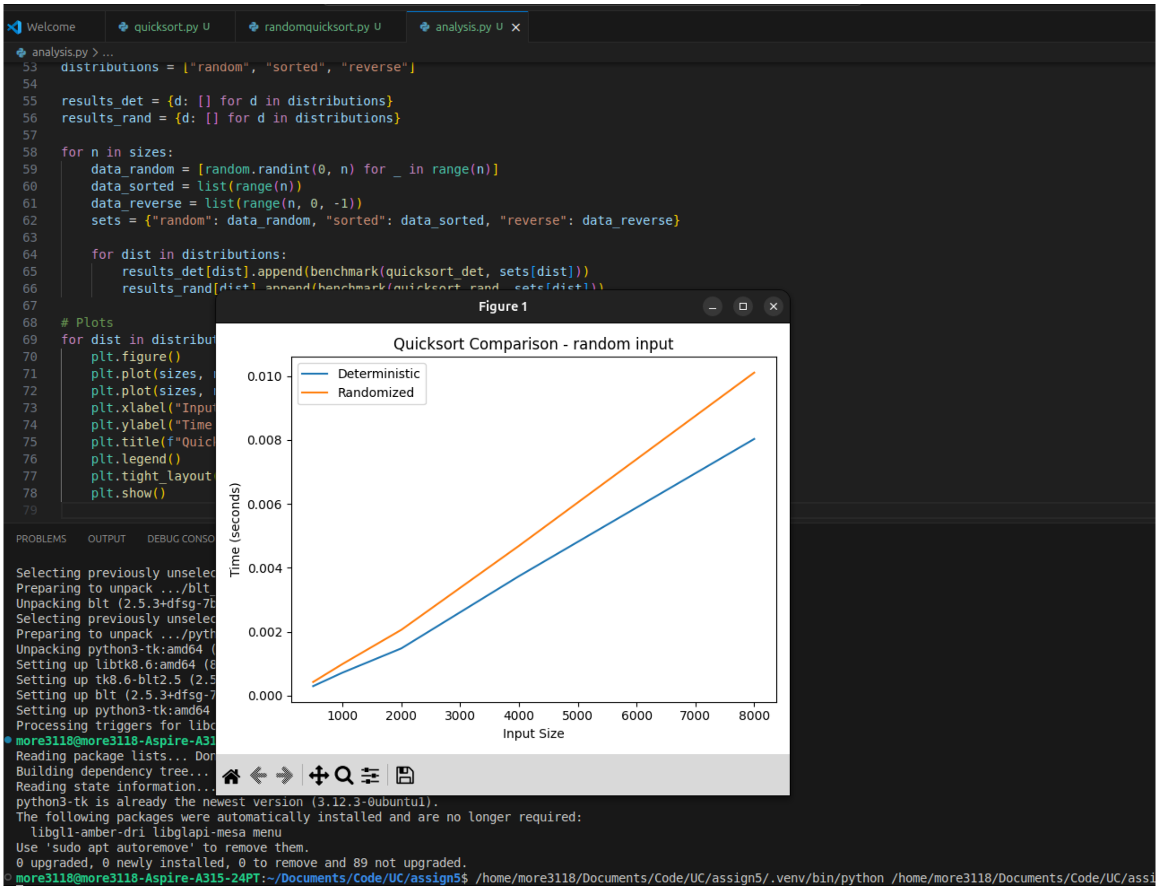
Install Matplotlib before proceeding. Use Virtual environment if required to run the programming



```
Sorted array: [1, 5, 7, 8, 9, 10]
⊗ more3118@more3118-Aspire-A315-24PT:~/Documents/Code/UC/assign5$ /bin/python3 /home/more3118/Documents/Code/UC/assign5/analysis.py
Traceback (most recent call last):
  File "/home/more3118/Documents/Code/UC/assign5/analysis.py", line 5, in <module>
    import matplotlib.pyplot as plt
ModuleNotFoundError: No module named 'matplotlib'
⊗ more3118@more3118-Aspire-A315-24PT:~/Documents/Code/UC/assign5$ py install matplotlib
Command 'py' not found, but can be installed with:
sudo apt install pythonpy
⊗ more3118@more3118-Aspire-A315-24PT:~/Documents/Code/UC/assign5$ pip install matplotlib
error: externally-managed-environment

× This environment is externally managed
╰─> To install Python packages system-wide, try apt install
    python3-xyz, where xyz is the package you are trying to
    install.
```

Result of comparison (Empircal Analysis)



```
distributions = ["random", "sorted", "reverse"]

results_det = {d: [] for d in distributions}
results_rand = {d: [] for d in distributions}

for n in sizes:
    data_random = [random.randint(0, n) for _ in range(n)]
    data_sorted = list(range(n))
    data_reverse = list(range(n, 0, -1))
    sets = {"random": data_random, "sorted": data_sorted, "reverse": data_reverse}

    for dist in distributions:
        results_det[dist].append(benchmark(quicksort_det, sets[dist]))
        results_rand[dist].append(benchmark(quicksort_rand, sets[dist]))

# Plots
for dist in distribut
    plt.figure()
    plt.plot(sizes,
    plt.plot(sizes,
    plt.xlabel("Input
    plt.ylabel("Time
    plt.title(f"Quic
    plt.legend()
    plt.tight_layout
    plt.show()
```

```
Selecting previously unselec
Preparing to unpack .../blt_
Unpacking blt (2.5.3+dfsg-7b
Selecting previously unselec
Preparing to unpack .../pyth
Unpacking python3-tk:amd64 (
Setting up libtk8.6:amd64 (8
Setting up tk8.6-blt2.5 (2.5
Setting up blt (2.5.3+dfsg-7
Setting up python3-tk:amd64
Processing triggers for libc
● more3118@more3118-Aspire-A31
Reading package lists... Don
Building dependency tree...
Reading state information...
python3-tk is already the newest version (3.12.3-0ubuntu1).
The following packages were automatically installed and are no longer required:
  libgl1-amber-dri libglapi-mesa menu
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 89 not upgraded.
○ more3118@more3118-Aspire-A315-24PT:~/Documents/Code/UC/assign5$ /home/more3118/Documents/Code/UC/assign5/.venv/bin/python /home/more3118/Documents/Code/UC/assi
```

Version	Average-case	Worst-case	Worst-case likelihood	Notes
Deterministic	O(n \log n)	O(n^2)	Moderate on	Very sensitive to input

Version	Average-case	Worst-case	Worst-case likelihood	Notes
pivot			structured inputs	order
Randomized pivot	O(n \log n)	O(n^2)	Extremely low	Hard to engineer worst-case
Median-of-three	O(n \log n)	O(n^2)	Low	Heuristic, not proof-level protection

Further Quicksort Time and Space Complexity Analysis

1. Best Case: O(n log n)

In the best case, the pivot divides the array into two nearly equal halves. The recurrence T(n) = 2T(n/2) + O(n) solves to O(n log n).

2. Average Case: O(n log n)

On average, partitions tend to be reasonably balanced. Solving the expected recurrence leads to O(n log n).

3. Worst Case: O(n^2)

Occurs when the pivot always produces maximally unbalanced partitions. The recurrence T(n)=T(n−1)+O(n) expands to O(n^2).

4. Space Complexity

Average: O(log n) due to recursion stack.

Worst: O(n) with highly unbalanced recursion.

Quicksort is in-place aside from recursion overhead.

References

GeeksforGeeks. (n.d.). Time and space complexity analysis of quick sort.

Scaler Topics. (n.d.). Quick sort algorithm.

Sourabh, S. K., & Chakraborty, S. (2008). How robust is quicksort average complexity? arXiv.

UMD Math / CMSC 351 Notes. (n.d.). QuickSort analysis.