# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2023** |
|  | | | | | |
| Points |  | Document author: | **Pratham Aggarwal** | | |
|  | Author’s email: | **Pratham\_aggarwal@mail.fresnostate.edu email** | | |
| Laboratory number: | **Lab 6** | | |
|  | | | | | |

**1. Statement of Objectives**

In this experiment, the Quick Sort algorithm was implemented with three different pivot options (the first element, a random element, and the median of the first, middle, and last element), and the execution times for sorting the array was analyzed. The approach, the results, and analysis of the Quick Sort algorithm with various pivot options are covered in this report.

**2. Experimental Procedure**

The experimental procedure involved implementing the Quick Sort algorithm with the following pivot choices:

Pivot Choice 1: The first element in the list.

Pivot Choice 2: A random element in the array.

Pivot Choice 3: The median of the first, middle, and last elements in the array.

I created a large array of 100,000 random numbers. The array was sorted recursively using the Quick Sort method after the selected pivot was swapped with the first element in each partition.

**3. Analysis**

The experimental results are as follows:

Execution Time with Pivot Choice 1 (First Element): 57 milliseconds

Execution Time with Pivot Choice 2 (Random Element): 52 milliseconds

Execution Time with Pivot Choice 3 (Median): 61 milliseconds

Analysis:

Pivot Choice 1 (First Element): This choice performed well, taking 57 milliseconds. When the array is already sorted, the worst-case time complexity is O(n^2).

Pivot Choice 2 (Random Element): This choice performed similarly, taking 52 milliseconds. Avoiding worst-case scenarios is made possible by selecting the pivot at random.

Pivot Choice 3 (Median of Three Elements): This choice took 61 milliseconds, which is a little longer. By taking into consideration the median of the first, middle, and last items, it attempts to improve pivot selection. Its time complexity is more like to O (n log n).

According to the results, random pivot selection (Choice 2) outperformed the other two options for this specific dataset. However, as I ran several times, sometimes the choice 3 outperformed the other two. There isn't much of a difference in execution time.

**4. Encountered Problems**

Throughout the lab, I ran into a number of difficulties. First, I had trouble writing the Quick Sort algorithm's pseudocode and implementing some predefined functions. For that, I took help from some online websites, lecture notes and tutors.

Another problem I ran across was that for short arrays, the execution time would always report as 0 milliseconds. To fix this, I expanded the array's size to 100,000 elements in order for a measurable execution time.

**5. Conclusions**

In conclusion, this experiment shows the Quick Sort algorithm's effects on sorting an array using three alternative pivot options. The algorithm's efficiency depends on the pivot choice. The median of three elements makes an attempt to further optimize pivot selection, but random pivot selection frequently offers good performance on average.

Key takeaways:

An effective sorting algorithm, Quick Sort has an O(n log n) average time complexity.

The algorithm's performance is significantly impacted by the pivot selection for much larger arrays.

**6. References**

Lecture Notes

