# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2021** |
|  | | | | | |
| Points |  | Document author: | **Marlon Branham** | | |
|  | Author’s email: | **zekebranham@mail.fresnostate.edu** | | |
| Laboratory number: | **Laboratory 2** | | |
|  | | | | | |

**1. Statement of Objectives**

In this lab, we are to create a program that implements a Quicksort algorithm. With this algorithm, comparing different pivot locations: the first element, a random element, and the median element. After creating the algorithm, compare the different pivot methods. The significance of this lab is learning which pivot method is the most efficient. In this report, I will be discussing the experimental procedure, analysis of results, encountered problems, and a conclusion.

**2. Experimental Procedure**

In this lab, we are to create a Quicksort algorithm with multiple pivot methods. I created two functions, one being my *partition* function, and the other being my actual *quickSort* function. In my *partition* function, it takes in of course the array, the size of the array, the low/high position of the array, and lastly, it takes in a partition choice element. Depending on which number (1-3) the user enters, determines the partition method used, being either the first element, a random element, or the median of the array, 1-3 respectively. The *quickSort* function recursively calls itself twice for both sides of the partitioned array after calling *partition* with the user-inputted choice of pivoting.

A computer screen shot of a program code

Description automatically generated

**3. Analysis**

The results of the implemented algorithm were too close to have a true conclusive answer since my computer can only handle a maximum of 500,000 elements. From my results, pivoting at the first element averaged 0.0825 seconds, pivoting at the median averaged 0.0877 seconds, and at a random element averaged 0.0956 seconds. My results show that the first element was the quickest partitioning for my case scenario. I believe that pivoting at the median would be the overall most efficient method since both sub-arrays will be close to if not equal in size. I also believe that using a random position would be the most effective since it eliminates worst and best-case situations being random. Overall, the choice of pivoting would vary from situation to situation.







**4. Encountered Problems**

The only issues I faced were regarding finding a large enough input size to have true variation in results. My computer can only handle 500,000 elements before it overflows, and I am not sure if there is anything I can do about that or if it is just a hardware issue. Since my array size was only 500,000, I would be able to get results, but I do wish I could experiment with a larger array.

**5. Conclusions**

In this lab, I learned the proper implementation of a Quicksort algorithm with both the partition function and the Quicksort function. I also learned that the method of pivoting relies more on the data you use, certain situations may require a certain pivot for better functionality. The last thing I learned is that my computer cannot handle more than 500,000 elements, I noticed this issue before but always looked past it. Overall, a successful lab working with Quicksort and different pivot methods.

**6. References**

<https://www.geeksforgeeks.org/quick-sort/>