# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2021** |
|  | | | | | |
| Points |  | Document author: | **Saishnu Ramesh Kumar** | | |
|  | Author’s email: | **saishnu\_rk@mail.fresnostate.edu** | | |
| Laboratory number: | **Section 1, 11am to 12:50pm** | | |
|  | | | | | |

**1. Statement of Objectives**

This lab assignment dealt with creating a quicksort function for three different pivots: first element as pivot, random element as pivot, and median as pivot. The time complexity for quicksort is O(nlogn). We are also asked to record the execution time for each pivot as they would vary depending on where the pivot is located to start the quicksort. The significance of this program is to understand how quicksort works as it is also another variation of the divide and conquer algorithm, which is similar to merge sort but in this case, this algorithm also requires a pivot that helps with the sorting.

**2. Experimental Procedure**

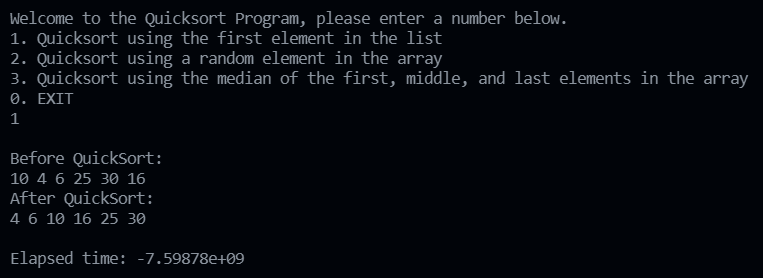
Since we were meant to have the quicksort function tackle three different pivots, I decided to create three different functions for each pivot and called them into their respective quicksort functions. I also created a swap function to swap the numbers around if needed. A partition function was also created for this as quicksort requires a partition function to help with updating the i and j in the array list properly. The quicksort function would basically tackle the sorting section where the partition function is called and then the array list would be split so that the quicksort function can be called recursively for each side of the list. Once the sorting is done, the printArray function would print out the sorted list of elements in the main function.

**3. Analysis**

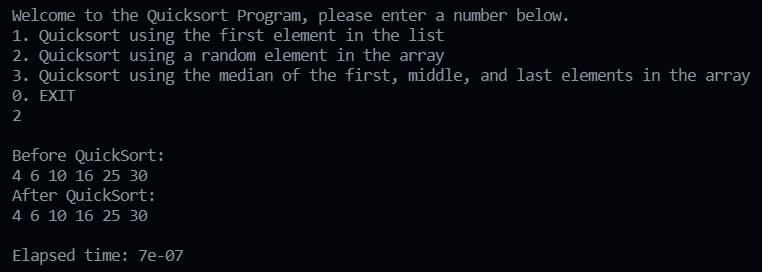
|  |  |  |
| --- | --- | --- |
| **Pivots** | **Time Complexity** | **Execution Time (seconds)** |
| First Element | O(nlogn) | **0.00000000759878** |
| Random Element | O(nlogn) | 0.0000007 |
| Median | O(nlogn) | 0.0000006 |

The time complexity for quicksort is O(nlogn). As for the execution time, it can be seen that first element is the fastest, followed by the median, and lastly the random element. It can be seen that if you were to set the lowest element as it’s pivot, the program would execute substantially fastest as compared to the other two forms of pivots that were given for this assignment.

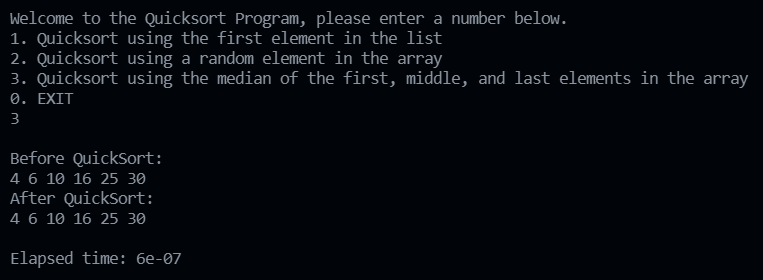
First Element Terminal Screenshot:



Random Element Terminal Screenshot:



Median Element Terminal Screenshot:



**4. Encountered Problems**

I managed to get the program to compile and run but I kept coming across Dumping Stack Errors which required me to go back through the code and attempt to debug and test each section of the code. I have also came across a couple of compiler errors but I was able to fix them and get the program compiling and running again.

**5. Conclusions**

I have learnt and got a better understanding about how quicksort works now. I have also learned the concept of having a partition being used for quicksort as that was not used in the previous divide and conquer algorithm, mergesort. Moreover, I have observed that first element is the fastest among the three pivots that were used for this program.

**6. References**

Slides provided during the lab class.

<https://www.softwaretestinghelp.com/quick-sort/#C_Example>

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

<https://cs.stackexchange.com/questions/99804/quick-sort-with-first-element-as-pivot>

<https://stackoverflow.com/questions/19417826/passing-function-pointer>