**CMPSC-462 –Data Structures and Algorithms**

**PROJECT-2**

**Goal:** The goal of this project is to perform search and sorting algorithms on an array/list and perform Time Complexity analysis on each function.

Write a program to perform Search and Sorting on randomly generated 10,000 and 50,000 integers.

1. Generate at least 10000 random numbers and store those in a list (name the list as Pool).
2. Generate a random number and call it as Target (use random.randint()).
3. Perform Linear Search and Binary Search to find Target in Pool. write each as a function.

Note: Binary search can be performed on sorted integers. Perform a sorting (call a sorting function) before calling binary search.

1. Perform Max/Min number search on the unsorted list. write each as a function.
2. Find/Search whether the unsorted list have distinct numbers. If it have distinct numbers return true and return false otherwise. If false, print the numbers which are not distinct. write it as a function.
3. Calculate the time taken for each function to perform the search. Note: for binary search, do not include the time taken for sorting the list.
4. Perform the following sorting on the Pool (write each as a function):

* Selection Sort
* Insertion Sort
* Bubble Sort
* Merge Sort.

**Note (Very Important):** perform these sorting on the unsorted list. Once you perform a sorting, the list would be sorted/ordered. Do not apply another sorting on the same list. So, before applying a sorting, make a copy of the unsorted list and apply sorting on this unsorted list. You can write a simple for loop or use inbuilt function to create a copy of the list.

1. Calculate the time taken for each sorting functions. If you get zero as recorded time, you can increase the number of elements to be sorted.

**Note:**

* Create a class file and call it as SearchClass. SearchClass should contain the search functions including max/min and distinct functions.
* Create another class file and call it as SortingClass. SortingClass should contain all the sorting functions.
* Call these appropriate class in your main class/function to perform the operations.
* All the required algorithms are covered in your lecture notes.

1. Perform these operations 3 times i.e. apply these functions on randomly generated numbers 3 times. Tabulate the results.
2. Continue the steps 1 – 9 for 50,000 random numbers. Tabulate the results.

**Report:** The project report should contain the following:

1st page: Project Title, course name, student’s name, instructor name and date

2nd page: Outline of the report with respective page numbers.

1. Introduction
2. Background

Write an algorithm for each function and explain the order of growth for each function i.e. calculate the Big O

1. Results/Sample Outputs

Note: If you use a large set of data (e.g. 10,000 or 50,000 numbers), you can show a part of the screenshot for your sample result. You need not show all the numbers in the sample.

1. Time Complexity Analysis

Tabulate the time taken for all the functions for 10,000 random integers.

Tabulate the time taken for all the functions for 100,000 random integers.

write a paragraph of your understanding.

1. Conclusion

What have you learnt from this project?

1. References
2. Appendix: (optional)

Program codes

**Due Date: 10/06/2022 for report submission (9:00 am)**

Total Points for Project-1: **100 points**

/\* All students are expected to use appropriate amount of comments to explain their program. \*/

Students should also be prepared to demonstrate the program during the lab session to the Instructor and as well as to the other students of the class if required.

**You should use the sample report format attached here.**



CMPSC 462: Data Structures and Algorithms (Fall 2022)

Project-2: Your Project Title

Student Name

Department

Instructor

Dr. Vinayak Elangovan, Ph.D.

Department of Computer Science

Submitted On: Date

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| 1 | INTRODUCTION | 1 |
| 2 | BACKGROUND | X |
| 3 | SAMPLE OUTPUTS | x |
| 4 | TIME COMPLEXITY ANALYSIS | x |
| 5 | CONCLUSION | X |
| 6 | REFERENCES | X |
| 7 | APPENDIX | x |

1. **INTRODUCTION**

Introduction for the project

// insert page numbers in all page and so you can refer it in the Outline.

1. **BACKGROUND**

Write an algorithm for each function and explain the order of growth for each function i.e. calculate the Big O

1. **SAMPLE OUTPUTS**

Note: If you use a large set of data (e.g. 10,000 or 100,000 numbers), you can show a part of the screenshot for your sample result. You need not show all the 10,000 or 100,000 numbers.

1. **TIME COMPLEXITY ANALYSIS**

Tabulate the time taken for all the functions and write a paragraph of your understanding.

1. **CONCLUSION**
2. **REFERENCES**
3. **APPENDIX**

Your program codes