Software Requirements Specification

For

Analyzing various types of searching and sorting algorithms

14th Sept, 2022

Prepared by

Specialization	SAP ID	Name
CCVT	500083967	Annie Jain
CCVT	500087335	Ashtosh Agarwal
CCVT	500082786	Ayush Juyal
CCVT	500082957	Harsh Goyal

Department of Systematics
School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

Table of Contents

1. INTRODUCTION

Let's imagine that we are in a big library and there are thousands and thousands of books in the shelves. Now we want to find the particular book of our interest through the library and the library is not sorted. Each and every book is placed at wrong place and order. How could we

sort to be able to search through to find the exact book we need? But if the library is well arranged i.e sorted then we could easily search our desired book from the library. So, we get to know how important is sorting and search in our real life. Same is in our computer science background. Searching and sorting plays very important role in creating the databases as we need to access it later on. Here in the project, we will be using various searching and sorting algorithms, showing you there practical implementation how they work, their efficiency and their properties. In technical language sorting basically helps to reduce the algorithmic complexity of the problem. A quick google search reveals that there are 40 different over 40 different sorting algorithms used on the computer science. On the other hand search is the process of finding the value in a list of values or it is the process of locating given value position in a list. Both the searching and sorting algorithms are of various types and are interesting too. Let's see how they work and their efficiency.

1.1 Purpose of the Project

The purpose of the project is to show the implementation of the various searching and shorting techniques and analyzing them. This project will tell us how various algorithm works in the complex database when we try to search the element throughout the or sort the contents. Since these searching and sorting are used in library, reception, bank, or another department so understanding it is more important. This will help to make the concepts of the data structure and algorithm stronger.

1.2 Target Beneficiary

The target beneficiary of this project is university student who are having problem in understanding the searching and sorting algo or any other who would like to see how the searching and sorting in a complex system works so smoothly.

1.3 Project Scope

The scope of the project is to tell about various searching and sorting. We do it every day in our daily life like: sorting our bank transaction, searching the traction of particular date, searching name in the university record or sorting alphabetically the names etc and these are very important to increase our efficiency, but we don't not know or how does it work or which type of algorithm it uses and to explain it this help. This project will tell us how algorithm work and their properties through the small database.

1.4 References

[1] Ajay Kumar, Bharat Kumar, Chirag Dawar and Dinesh Bajaj, Comparison Among Different Sorting Techniques, International Journal for Research In Applied Science And Engineering Technology (IJRASET), 2014
[2] Ramesh Chand Pandey, Study and Comparison of various Sorting Algorithms, 2008
[3] https://www.geeksforgeeks.org/sorting-algorithms/
[4] https://www.javatpoint.com/searching-in-binary-search-tree

2. PROJECT DESCRIPTION

2.1 Algorithm

Searching Algorithms-

- Linear Search
- Binary Search
- Exponential Search
- Fibonacci Search
- Interpolation Search
- Sub list Search
- Ubiquitous Binary Search
- Jump Search

Sorting Algorithm-

- Selection Sort
- Bubble Sort
- Quick Sort
- Merge Sort
- Insertion Sort
- Heap Sort
- Counting Sort
- Radix Sort
- Bucket Sort

2.2 SWOT Analysis

Strength: The utmost strength of our project is that it is a real-time problem-solving project and can help to analyze various searching and sorting algorithms. This project will tell us how various algorithm works in the complex database when we try to search the element throughout the or sort the contents.

Weakness: The only weakness of our project is to implement various algorithmic techniques at once.

Opportunities: It can have wide range of valid possibility that this project can evolve into some bigger scenarios. This project can also be used for saving our time in finding the number of searching and sorting algorithms here and there on the web and can explain you everything in a very quick and precise manner at a single place only.

Threats: The threat in our algorithm is to constantly update and add the new searching and sorting techniques which will be built in future.

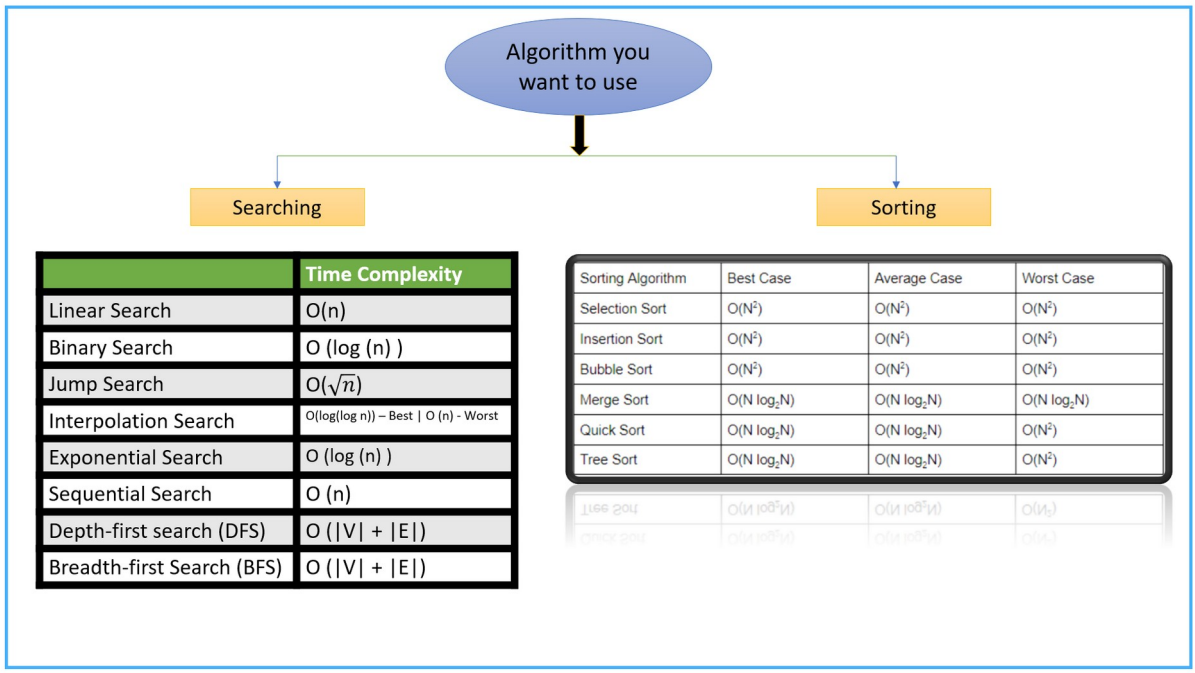
2.3 Project Features

The feature of the project is that it will help in sorting of data in ascending, descending order as well as in alphabetical order. And this project is also helpful in finding a particular data in the whole database easily.

2.4 Design and Implementation Constraints

- Complexity in integrating and comparing various searching and sorting techniques at the same time

2.5 Design diagram



	Time Complexity
Linear Search	O(n)
Binary Search	O (log (n))
Jump Search	$O(\sqrt{n})$
Interpolation Search	O(log(log n)) – Best | O (n) - Worst
Exponential Search	O (log (n))
Sequential Search	O (n)
Depth-first search (DFS)	O (|V| + |E|)
Breadth-first Search (BFS)	O (|V| + |E|)

Sorting Algorithm	Best Case	Average Case	Worst Case
Selection Sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
Insertion Sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
Bubble Sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
Merge Sort	$O(N \log_2 N)$	$O(N \log_2 N)$	$O(N \log_2 N)$
Quick Sort	$O(N \log_2 N)$	$O(N \log_2 N)$	$O(N^2)$
Tree Sort	$O(N \log_2 N)$	$O(N \log_2 N)$	$O(N^2)$

3. SYSTEM REQUIREMENTS

3.1 User Interface

A command-line interface (CLI) is a text-based UI used to run programmes, manage computer files, and interact with the computer. CLIs are also known as command-line user interfaces, console user interfaces, and character user interfaces. CLIs accept keyboard commands as input; today, most vendors provide the graphical user interface (GUI) as the default for

operating systems (OSes) such as Windows, Linux, and macOS. Most modern Unix-based systems provide both a command-line and a graphical user interface. Command-line interfaces can be found in the MS-DOS operating system and the command shell in the Windows operating system. Furthermore, command-line interfaces can be supported by programming language development platforms such as Python. For IT professionals, software developers, system administrators, network administrators, and others who prefer a more precise and reproducible interface to their systems, the command line remains an important tool.

3.2 Algorithm

Searching Algorithms-

- Linear Search- *Linear Search is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found, otherwise the search continues till the end of the data set. It is the easiest searching algorithm*
- Binary Search- *Binary Search is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).*
- Exponential Search- Exponential Binary Search is particularly useful for unbounded searches, where size of array is infinite. Please refer Unbounded Binary Search for an example. It works better than Binary Search for bounded arrays, and also when the element to be searched is closer to the first element.

- Fibonacci Search- Fibonacci Search is a comparison-based technique that uses Fibonacci numbers to search an element in a sorted array.
- Interpolation Search- The Interpolation Search is an improvement over Binary Search for instances, where the values in a sorted array are uniformly distributed. Interpolation constructs new data points within the range of a discrete set of known data points. Binary Search always goes to the middle element to check. On the other hand, interpolation search may go to different locations according to the value of the key being searched. For example, if the value of the key is closer to the last element, interpolation search is likely to start search toward the end side.
- Sub list Search- Sublist search is used to detect a presence of one list in another list. Suppose we have a single-node list (let's say the first list), and we want to ensure that the list is present in another list (let's say the second list), then we can perform the sublist search to find it.
- Ubiquitous Binary Search- Here we need log N + 1 comparisons in worst case. If we observe, we are using two comparisons per iteration except during final successful match, if any. In practice, comparison would be costly operation, it won't be just primitive type comparison. It is more economical to minimize comparisons as that of theoretical limit.
- Jump Search- Like Binary Search, Jump Search is a searching algorithm for sorted arrays. The basic idea is to check fewer elements (than linear search) by jumping ahead by fixed steps or skipping some elements in place of searching all elements.

Sorting Algorithm-

- Selection Sort- The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning.
- Bubble Sort- Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

- Quick Sort- QuickSort is a Divide and Conquer algorithm. It picks an element as a pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.
  - Always pick the first element as a pivot.
  - Always pick the last element as a pivot (implemented below)
  - Pick a random element as a pivot.
  - Pick median as the pivot.
- Merge Sort- The Merge Sort algorithm is a sorting algorithm that is based on the Divide and Conquer paradigm. In this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner.
- Insertion Sort- **Insertion sort** is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

- Heap Sort- *Heap sort is a comparison-based sorting technique based on Binary Heap data structure. It is similar to the selection sort where we first find the minimum element and place the minimum element at the beginning. Repeat the same process for the remaining elements.*
- Counting Sort- Counting sort is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values (kind of hashing). Then do some arithmetic to calculate the position of each object in the output sequence.
- Radix Sort- If we have $\log_2 n$ bits for every digit, the running time of Radix appears to be better than Quick Sort for a wide range of input numbers. The constant factors hidden in asymptotic notation are higher for Radix Sort and Quick-Sort uses hardware caches more effectively. Also, Radix sort uses counting sort as a subroutine and counting sort takes extra space to sort numbers.
- Bucket Sort- Bucket sort is mainly useful when input is uniformly distributed over a range. For example, consider the following problem.
  Sort a large set of floating point numbers which are in range from 0.0 to 1.0 and are uniformly distributed across the range. How do we sort the numbers efficiently?
  A simple way is to apply a comparison based sorting algorithm. The lower bound for Comparison based sorting algorithm (Merge Sort, Heap Sort, Quick-Sort .. etc) is $\Omega(n \log n)$, i.e., they cannot do better than nLogn.

4. NON-FUNCTIONAL REQUIREMENTS

4.1 Performance requirements

The following are the four basic requirements needed:

- Analyzing various algorithm
- properties various algorithm
- Software Updates and Maintenance

4.2 Software Quality Attributes

Availability: Our project is always available since the basic resources are used which are always available.

Efficiency- Our program is efficient as basic services are used which are very efficient.