ABSTRACT

Searching and Sorting is one of the most fundamental problems in Data, as it is use in most software applications. Data-driven algorithm is especially use sorting to gain an efficient access to data. Many sorting algorithms distinct properties for different architectures have been developed. Searching for items and Sorting through items are tasks that we do every day. Searching and sorting are also common task in computer programs. We search for all occurrences of a world in a file in order to replaces it with another word. We sort the items on list into alphabetical or numerical order. Because searching and sorting are common computer tasks, we have well known algorithms, for doing Searching and Sorting Algorithms like Boyer Moore Algorithm, KMP Algorithm, Raita Algorithm, Rabin karp Algorithm and Sunday Algorithm and these algorithms we are using in this project. We will look at this algorithm in details in our project. And we are comparing these algorithms with each other in terms of time complexity, space complexity, notations and determine the performance of each algorithms, in term of how quickly each algorithm completes its task.

Key Words: Boyer Moore Algorithm, KMP Algorithm, Raita Algorithm, Rabin Karp Algorithm and Sunday Algorithm.

ACKNOWLEDGEMENTS

We would like to express our deepest appreciation to all who provided us the possibility to do this project report. We are grateful to our principal **Dr K.S.Wani**. Furthermore, we thank to our guide **Prof.R.B.Sangore**, whose contribution in stimulating suggestions and encouragement, helped us to writing this report.

Furthermore, we would also like to acknowledge with much appreciation the Head of the department **Dr.U.S.Bhadade**. Then we would like to thank with crucial role of all staff member, who give the suggestion and the necessary material to complete the task.

We would like to thank all those who have contribute to the completion of the report and helped us with valuable suggestions for improvement. last but not least, we extremely indebted our parents and friends, without their support this effort had not reach its successful completion.

CONTENT

CHAPTER 1

INTRODUCTION

The searching and Sorting algorithm are used in many programs. There were many type of sorting and searching algorithm available in Data Structure.

Search is the process of finding the value in the list of values. In other words, Searching is the process of locating given value position in the lists of values.

Algorithm is sequence of instructions or set of rules that are follow to complete a task. In the discussion that follows, we use the term search term to indicate the Item for which we are searching. We assume the lists to search is an array of integers, although this algorithm will work just as any other primitive datatype.

Sorting refer to the operations of arranging data in some given sequence that i.e. increasing or decreasing order. Sorting is categorized as internal sorting and external sorting. Internal sorting means we are arranging the elements within the array which is only in computer primary memory. whereas external sorting is the sorting of elements from the external files by reading it from secondary memory.

In our program we are using the following searching and sorting algorithm:

1. Sunday Algorithm
2. Boyer Moore Algorithm
3. KMP Algorithm
4. Raita Algorithm
5. Rabin Karp Algorithm

In these searching and sorting algorithm we are comparing with each other in terms of time complexity, space complexity, notation.

1.**Sunday Algorithm**: The Sunday algorithm assumes its best case if every time in the first comparison a text symbol is found that does not occur at all in the pattern. Then the algorithm performs just $O(n/m)$ comparisons.

In contrast to the Boyer-Moore and the Horspool algorithm the pattern symbols need not be compared from right to left. They can be compared in an arbitrary order. For instance, this order can depend on the symbol probabilities, provided they are known. Then the least probable

symbol in the pattern is compared first, hoping that it does not match, so that the pattern can be shifted

2.Boyer Moore Algorithm : **Boyer–Moore string-search algorithm** is an efficient string-searching algorithms that is the standard benchmark for practical string-search literature. The algorithm pre-processes the string being searched for (the pattern), but not the string being searched in (the text). It is thus well-suited for applications in which the pattern is much shorter than the text or where it persists across multiple searches. The Boyer-Moore algorithm uses information gathered during the pre-process step to skip sections of the text, resulting in a lower constant factor than many other string search algorithms. In general, the algorithm runs faster as the pattern length increases. The key features of the algorithm are to match on the tail of the pattern rather than the head, and to skip along the text in jumps of multiple characters rather than searching every single character in the text.

3.KMP Algorithm: The **Knuth–Morris–Pratt string-searching algorithm** (or **KMP algorithm**) searches for occurrences of a "word" W within a main "text string" S by employing the observation that when a mismatch occurs, the word itself embodies sufficient information to determine where the next match could begin, thus bypassing re-examination of previously matched characters.

The algorithm was conceived by James H. Morris and independently discovered by Donald Knuth "a few weeks later" from automata theory.[1][2] Morris and Vaughan Pratt published a technical report in 1970. The three also published the algorithm jointly in 1977.[1] Independently, in 1969, Matiyasevich discovered a similar algorithm, coded by a two-dimensional Turing machine, while studying a string-pattern-matching recognition problem over a binary alphabet. This was the first linear-time algorithm for string matching.

4.Raita Algorithm : Raita designed an algorithm which at each attempt first compares the last character of the pattern with the rightmost text character of the window, then if they match it compares the first character of the pattern with the leftmost text character of the window, then if they match it compares the middle character of the pattern with the middle text character of the window. And finally, if they match it actually compares the other characters from the second to the last but one, possibly comparing again the middle character.

Raita observed that its algorithm had a good behaviour in practice when searching patterns in English texts and attributed these performances to the existence of character dependencies.

Smith made some more experiments and concluded that this phenomenon may rather be due to compiler effects.

5.**Rabin Karp Algorithm** : **Rabin–Karp algorithm** or **Karp–Rabin algorithm** is a string-searching algorithm created by Richard M. Karp and Michael O. Rabin (1987) that uses hashing to find an exact match of a pattern string in a text. It uses a rolling hash to quickly filter out positions of the text that cannot match the pattern, and then checks for a match at the remaining positions. Generalizations of the same idea can be used to find more than one match of a single pattern, or to find matches for more than one pattern.

To find a single match of a single pattern, the expected time of the algorithm is linear in the combined length of the pattern and text, although its worst-case time complexity is the product of the two lengths. To find multiple matches, the expected time is linear in the input lengths, plus the combined length of all the matches, which could be greater than linear. In contrast, the Aho–Corasick algorithm can find all matches of multiple patterns in worst-case time and space linear in the input length and the number of matches (instead of the total length of the matches).

CHAPTER 2

PROBLEM DEFINITION

Basic task in many types of computer applications. Especially when large amounts of data are to be sorted, efficiency becomes a major issue. There are many different sorting and searching algorithms and even more ways in which they can be implemented.

searching and sorting are common computer tasks, we have well known algorithms, for doing searching and sorting Algorithms like Boyer Moore Algorithm, KMP Algorithm, Raita Algorithm, Rabin karp Algorithm and Sunday Algorithm and these algorithms we are using in this project. We will look at this algorithm in details in our project.

When does an algorithm provide a satisfactory solution to a problem?

- One measure of efficiency is the time used by a computer to solve a problem using the algorithm, when input values are of a specified size
- second measure is the amount of computer memory required to implement the algorithm when input values are of a specified size

Questions such as these involve the **computational complexity** of the algorithm. An analysis of the time required to solve a problem of a particular size involves the **time complexity** of the algorithm. An analysis of the computer memory required involves the **space complexity** of the algorithm.

To eliminate the above issues, our project is going to use the concept of "In the current searching & sorting algorithms". Especially when large amounts of data are to be sorted, efficiency becomes a major issue. There are many different searching and sorting algorithms and even more ways in which they can be implemented. it involves the performance of algorithms, about their run time and space requirements. Analysing an algorithm is to discover its characteristics in order to evaluate its suitability for various applications.

CHAPTER 3

LITERATURE SURVEY

The modern era is the age of expertise and focuses on new development along with the comparing of existing technologies. Here, a comparative study aims to come up with the most efficient sorting algorithms or the techniques used. Sorting is an important data structure operation for managing data. Sorting algorithms and searching techniques are both distinct. Searching is based on searching the element for the given list, where in Sorting is to arrange the given list in a particular order which can be in ascending or descending order.

Most of these data are being compared, sorted only for a portion and the piece of data which is actually used for determining sorted order is known as the Key. The methodology used is to evaluate the performance using CPU time and memory space. Sorting algorithms and searching technique can operate both on sorted and unsorted list of elements. The items may be stored individually as records in a database or may be elements of a search space defined by a procedure.

There are different types of algorithms and techniques for performing different tasks and as well as same tasks, with each having its own advantages and disadvantages depending on the type of data structure. An analysis is being carried out on various sorting algorithms and searching techniques on parameters like space and time complexity. Based on the analysis, a comparative study is being made so that the user can choose the type of technique used based on the requirement.

CHAPTER 4

ANALYSIS

Algorithm analysis should begin with a clear statement of the task to be performed. This allows us both to check that the algorithm is correct and to ensure that the algorithms we are comparing perform the same task.

Although there are many ways that algorithms can be compared, we will focus on two that are of primary importance to many data processing algorithms:

- *Time complexity*: how the number of steps required depends on the size of the input
- *Space complexity*: how the amount of extra memory or storage required depends on the size of the input

In this algorithm, there is data structure, time complexity, space complexity and notation etc. all these things should be examining.

When we take this algorithm for analysis, we face the first thing that space complexity

SELECTION OF LANGUAGE: Selection of language was a critical thing for us because interface matters. Its interface is good than the things are able to access easily and fast so we can select and we decided to do in C/C++ language.

COMPILER: turbo c /gcc/Min GW Compiler

FONTEND: front end of system is C++.

OPERATING SYSTEM: WINDOWS/UBUNTU

STRUCTURE: structure should be understandable to the user. hence, we have made a structure a less complex so that user can easily access the things.

CHAPTER 5

DESIGN

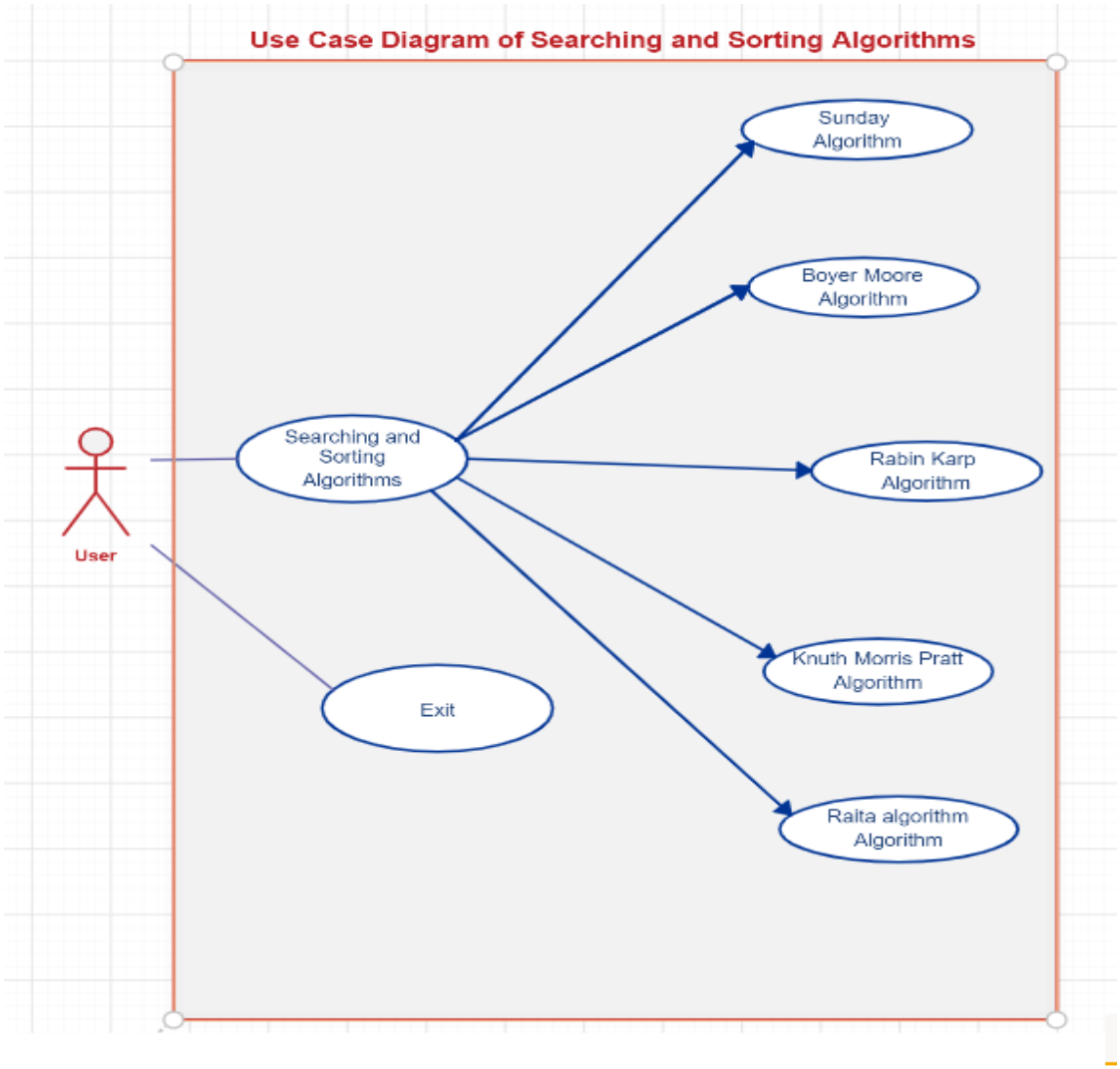

Fig. No.1 Use case of Searching and Sorting Algorithms

- A use case is a list of actions or event steps typically defining the interactions between a role of an actor (User) and a system to achieve a goal. A use case is a useful technique for identifying, clarifying, and organizing system requirements.
- In user (actor) gives the input (string &sub-string)select a techniques algorithms (searching and sorting) then it perfrom operations in algorithms(Sunday,boyer moore,rabin karp, kmp and raita algorithms).

DATAFLOW DIAGRAM: LEVEL 0

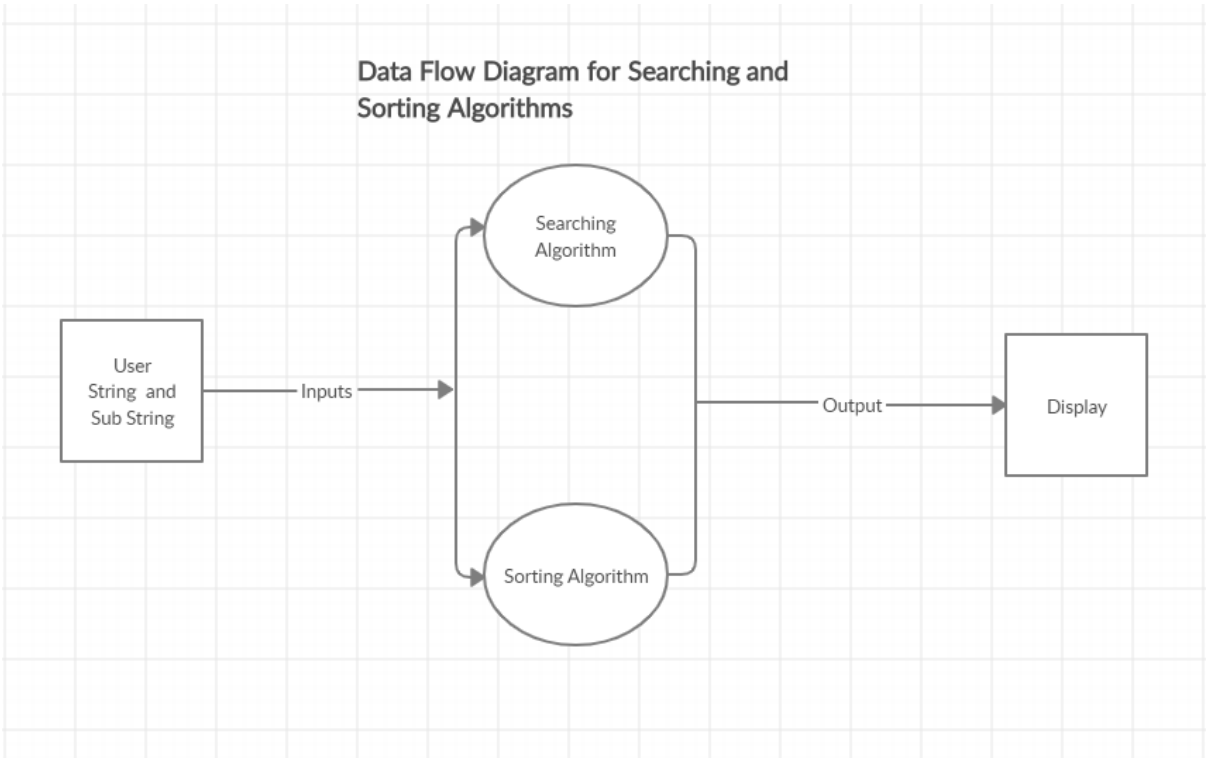

Fig. No.2. DFD 0 Level of Searching and Sorting Algorithms

- In user (Entite) gives the input (string &sub-string) select a techniques algorithms (searching and sorting) then it sort or search lengh it provide output

DATAFLOW DIAGRAM: LEVEL 1

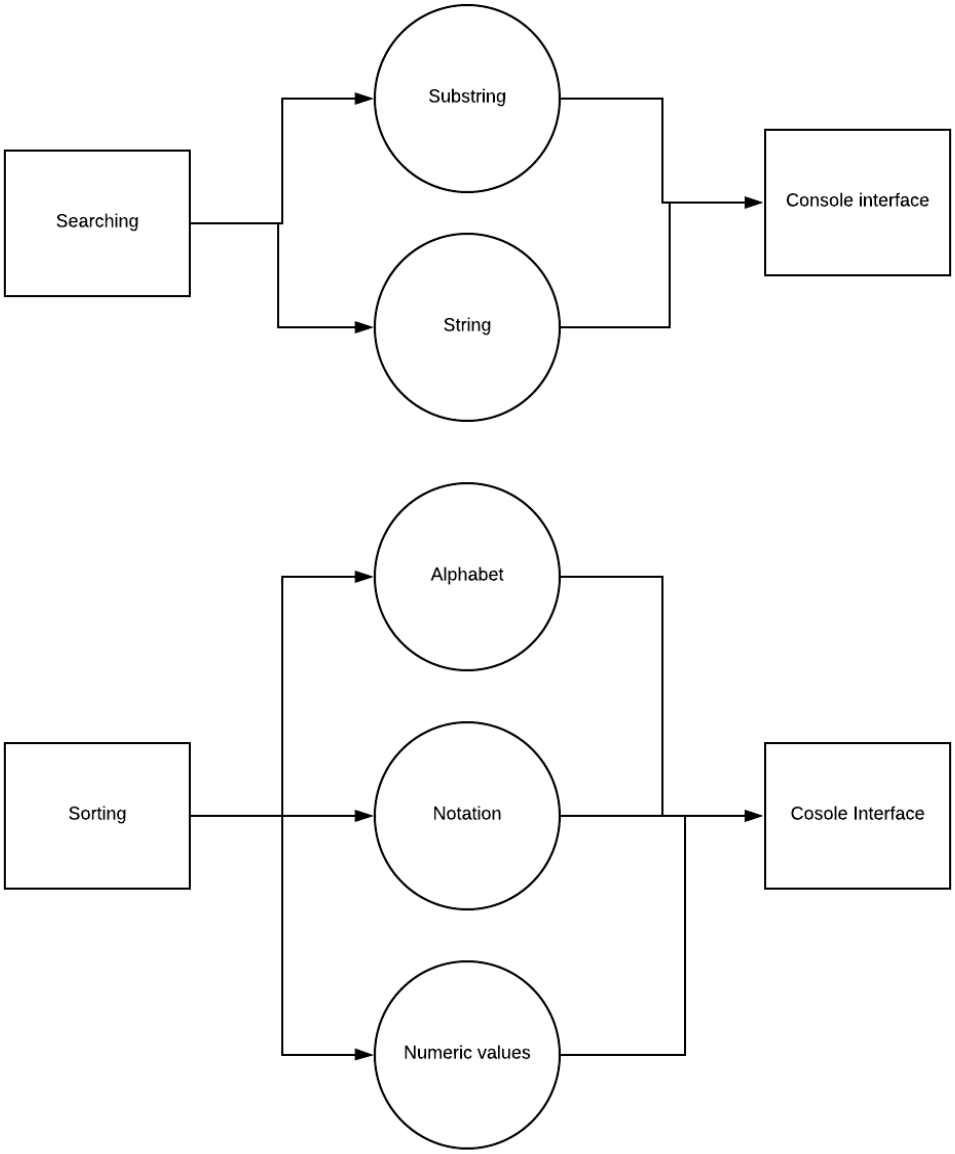

Fig. No.3 DFD 1 Level of Searching and Sorting Algorithms

- In user (Entites) gives the input (Searching) then it perfrom operation such as (string and substring) provide output (console interface).
- In user (Entites) gives the input (Sorting) then it perfrom operation such as (Alphabets,Nootation and Numeric values) provide output (console interface).

CLASS DIAGRAM

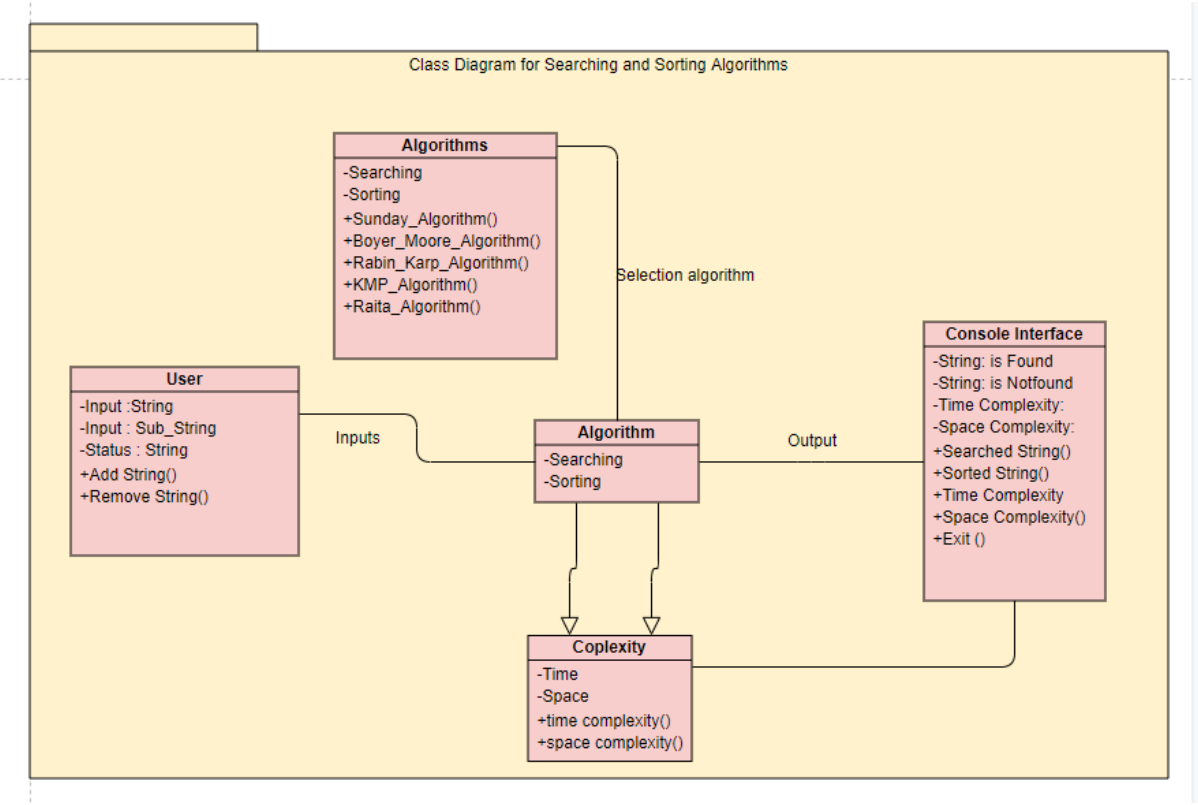

Fig. No.4 Class Diagram of Searching and Sorting Algorithms

References:

- Data Structures and Algorithms Made Easy by Narasimha Karumanchi
- INTRODUCTION TO ALGORITHMS THIRD EDITION Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
- Advanced Algorithmic Trading by Michael L. Halls-Moore