FINDING WORDS IN ASCENDING OR DESCENDING ORDER

Swaraj Bhosle (IIT2019024)
Ritesh Raj (IIT2019025)
Utkarsh Garg(IIT2019026)

*4th Semester B.Tech , Department of Information Technology,
IIIT, Allahabad, India*

Abstract: *In this Paper we have devised an algorithm to find the number of valid english words which have ascending(A to Z) or descending(Z to A) ordered characters*

I. INTRODUCTION

In computer science, the process of searching a string in a given set of strings is known as String Searching. We have created a text file containing at least 1000 valid english words. Basically ascending characters in a string means all the characters should be in a alphabetical order i.e from A to Z and the descending order means all the characters should be in a reverse alphabetical order i.e from Z to A. For creating a new algorithm from scratch, we look into the many crucial aspects of algorithm designing and analysis.The algorithm created is tested for different sample test cases for time complexity, space complexity and accuracy. In our case

This report further contains -

II. Algorithm Description
III. AlgorithmAnd Analysis

IV. Time Complexity Analysis
V.Conclusion

.

II. ALGORITHM DESCRIPTION

We are using a file system in our algorithm as our input is stored in the file. Our algorithm will first of all read the input from the txt file.
We have created the "input.txt" file and all the input strings are inside this file.

Step-1: First thing we will open the "input.txt" file and will read the input for the programme.

To read the information from a file into your programme we have to use the extraction operator (>>) just as we use that operator to input information from the keyboard. The only difference is that we use an ifstream or fstream object instead of the cin object.

Step-2: check whether the file is opened or not. i.e the open fails if the file doesn't exist, or if it can't be read. a failure can be detected with the code like !(logical not) operator

Step-3: create the character array of MAX_SIZE of 1000. We will store the input string from the file into this character array.

Step-4: Traverse through each input string from .txt file .

Step-5: create two different character arrays and store the input string in these two character arrays differently.

Step-6: Sort one of the above taken arrays in alphabetical order (A to Z) and sort the other array in reverse alphabetical order (Z to A).

Step-7: compare the original string with these two created ascending and descending strings. If the original string matches one of the two strings then count it.

Step-8: print the count number.

9. close the opened file

Advantages of using merge sort:-

1) It is quicker for larger lists because unlike insertion and bubble sort it doesn't go through the whole list several times.
2) It has a consistent running time, carries out different bits with similar times in a stage.

III. ALGORITHM AND ANALYSIS

Input : Taking string input from the text document.

Output result: The number of English words that have ascending or descending characters.

Pseudocode :

Function **Main()**{

   //Declare an input file steam

 ifstream inFile;

   //open the file stream
 inFile.open("input.txt");

 //Check that the file was opened

if (file can't open)

      Exit the code

//create the char array of MAX_SIZE

      char str[MAX_SIZE]

//assign count equal to zero

 count =0

```
while(inFile >> str) {

  //Store the length of string in l

    n = strlen(str)

//create two char array of MAX_SIZE

  char  asc[MAX_SIZE]
  char  dsc[MAX_SIZE]

 //store str string in both asc and dsc array

For( i=0 to i=n){
    asc[i] = str[i];
    dsc[i] = str[i]
}

//Sort the asc char array in ascending order

merge_sort(asc, asc + l)

//Sort the dsc array in descending order

merge_sort(dsc, dsc + l, greater<char>());

//compare the str string with dsc and asc array.

if(strcmp(str, asc) == 0 || strcmp(str, dsc) == 0)

        count++;

//if it matches any one of them then count it.

}

//print the count

  cout<<count

 //close the file

  inFile.close();
```
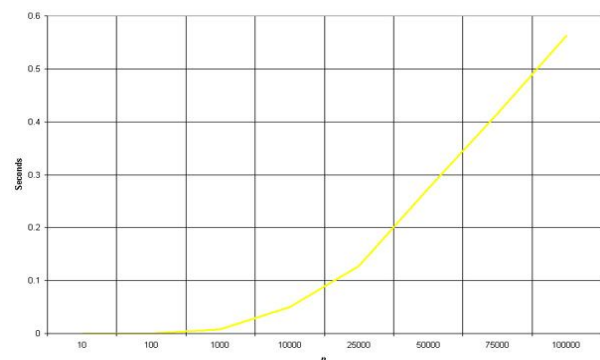
```
}
```


IV. TIME CALCULATION

No. of words	Execution Time(ms)
1000	0.31939
10000	0.52023
100000	0.87831
1000000	1.49021

V. TIME COMPLEXITY

Let the average size of all strings be N.

Then we merge sort the string in ascending and descending order and compare the obtained strings with the original string, and if the string matches with any of the sorted strings then we increment the answer.

The time complexity will be O(NlogN) due to the merge sort algorithm in a best, average and worst case.

The time complexity will be affected due to the merge sort in sorting the 1D array and the total time complexity will become $O(N*log(N))$ in all the cases ie. best case, worst case and average.

Best case :
$O(N) + O(N*log(N)) + O(N) => O(N*log(N))$
Average case :
$O(N) + O(N*log(N)) + O(N) => O(N*log(N))$
Worst case :
$O(N) + O(N*log(N)) + O(N) => O(N*log(N))$

BEST	AVERAGE	WORST CASE
N*log(N)	N*log(N)	N*log(N)

VI. CONCLUSION

Merge sort is one of the most efficient sorting algorithms. It works on the principle of Divide and Conquer. Merge sort repeatedly breaks down a list into several sublists until each sublist consists of a single element and merging those sublists in a manner that results into a sorted list in logarithmic time.

This algorithm works as the string gets passed through the sorting algorithm and sorted in ascending and descending order which helps in comparing with the initial string of the text document and calculating the number of english words that are in ascending or descending characters.

VII. REFERENCES

1.http://www.fredosaurus.com/notes-cpp/io/readtextfile.html
2.https://www.tutorialspoint.com/cplusplus/cpp_files_streams.html
3.https://www.geeksforgeeks.org/program-sort-string-descending-order