

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JNANA SANGAMA”, BELAGAVI-590018, KARNATAKA



A Mini Project Report On

“DOCUMENT PROCESSING USING KEYWORDS”

Submitted in the partial fulfilment of the requirement for the completion of **File Structure Laboratory with Mini Project (18ISL67)** and award of degree of

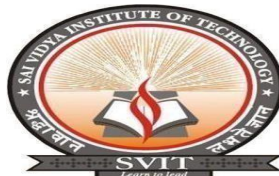
**BACHELOR OF ENGINEERING IN
INFORMATION SCIENCE AND ENGINEERING**

Submitted By

Deepika	1VA18IS006
Chola Deepthi	1VA17IS017
Priyanka D V	1VA18IS024

Under the Guidance of

Mr. Abhijit Das
Assistant Professor
Dept. of ISE, SVIT



DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

SAI VIDYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of
Karnataka | Approved by AICTE, New Delhi)

RAJANUKUNTE, BENGALURU – 560 064

2021-22

SAI VIDYA INSTITUTE OF TECHNOLOGY

Rajankunte, Bengaluru- 560 064

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the Mini project work entitled "**DOCUMENT PROCESSING USING KEYWORDS**" carried out by **Ms. Deepika (1VA18IS006)**, **Ms. Chola Deepthi (1VA17IS017)**, **Ms. Priyanka (1V18IS024)** students of **SAI VIDYA INSTITUTE OF**

TECHNOLOGY, Bengaluru, in partial fulfilment for the award of Bachelor of Engineering in Information Science and Engineering of **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**, Belagavi during the year **2021-22**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of mini-Project work prescribed for the **File Structure Laboratory with Mini Project(18ISL67)**.

Mr. Abhijit Das

Assistant Professor,

Dept. of IS&E, SVIT

Dr. Vrinda Shetty

HOD

Dept. of IS&E, SVIT

Dr. H S Ramesh Babu

Principal

External Viva:

Name

Signature

1. _____

2. _____

ABSTRACT

A file structure is a combination of representation of data in files and operations for accessing the data. It allows applications to read, write and modify data. The project titled “Document Processing Using Keyword”, where it helps to search keywords in a text file. In this process, the large text file is exposed to perform operations like search, modify and indexing where search helps to find keywords in a file, modify helps in altering the information and indexing helps to find the exact position of the words in the text file. The purpose of this project is to reduce the time consumed to perform these operations by the help of computerized equipments and full-fledged computer software fulfilling their requirements.

ACKNOWLEDGEMENT

The completion of the mini project brings with a sense of satisfaction, but it is never complete without thanking the persons who are all responsible for its successful completion. First and foremost, I wish to express our deep sincere feelings of gratitude to my Institution, **Sai Vidya Institute of Technology**, for providing us an opportunity to do our education.

I would like to thank the **Management, Prof. M R Holla**, Director, Sai Vidya Institute of Technology and **Prof. A M Padma Reddy**, Director (A), Sai Vidya Institute of Technology for providing the facilities.

I extend my deep sense of sincere gratitude to **Dr. H S Ramesh Babu**, Principal, Sai Vidya Institute of Technology, Bengaluru, for having permitted to carry out the project work on “**DOCUMENT PROCESSING USING KEYWORDS**” successfully.

I express my heartfelt sincere gratitude to **Dr. Vrinda Shetty**, HOD, Department of Information Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for her valuable suggestions and support.

I express my special in-depth, heartfelt, sincere gratitude to **Mr. Abhijit Das**, Assistant Professor, Department of Information Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for their constant support.

Finally, I would like to thank all the Teaching, Technical faculty and supporting staff members of Department of Information Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for their support.

Deepika	1VA18IS006
Chola Deepthi	1VA17IS017
Priyanka D V	1VA18IS024

TABLE OF CONTENTS

ABSTRACT	1
ACKNOWLEDGEMENT	2

Chapter No	Chapter Name	Page No
1	INTRODUCTION	1-4
2	LITERATURE SURVEY	5-6
3	REQUIREMENT ANALYSIS AND DESIGN	7-8
4	SYSTEM MODELING	9
5	IMPLEMENTATION	10-11
6	SNAPSHOTS	12-18
7	FUTURE WORK AND CONCLUSION	19
	REFERENCE	20

CHAPTER 1

INTRODUCTION

1.1 File Structure

A file structure is the organization of data in files and files in secondary storage and operations for accessing the data. A file structure allows applications to read, write and modify data. It also supports in finding the data that matches some search criteria.

Earlier files were stored on tapes. Access to these tapes was sequential. The cost to access the tape grew in direct proportion to the size of the file.

As files grew and as storage devices such as disk drives became available, indexes were added to files. Index consist of a list of keys and corresponding pointers to the files. This type of accessing files could search the files quickly.

As the index file grew, they too became difficult to manage. In early 1960's, the idea of applying tree structures emerged. But this also resulted in long searches due to uneven growth of trees, as a result of addition & deletion of records.

In 1963 researchers developed, an elegant, self-adjusting binary tree structure called the AVL tree, for storing data in memory. The problem with this binary tree structure was that dozens of accesses were required to find a record.

The solution to this problem emerged in the form of B-tree. B-tree grows from the bottom-up, as records are inserted. B-tree provided excellent access performance, except that the files couldn't be accessed sequentially. This problem was solved by introducing B+ trees.

But all the above methods needed more than one disk access. Hashing approach promised to give required data in one disk access, but performed badly for large amount of data.

Extendible Dynamic hashing approach is proposed to improve the performance

of hashing algorithm.

1.1 File Access Operations:

Seek time: The time taken to move the read/write head to the required cylinder, is called seek time. The amount of seek time spent depends on how far the read/write head has to move. If we are accessing a file sequentially and the file is packed into several consecutive cylinders, then the seek time required for consecutive access of data is less. Since the seek time required for each file operation varies, usually the average seek time is determined. Most hard disks available today have an average seek time of less than 10 milliseconds.

Rotational delay: The time taken for a disk to rotate and bring the required sector under read/write head is called rotational delay. On average, this is considered to be half of the time taken for one rotation. If a hard disk rotates at 5000rpm, then it takes 12msec to complete one rotation, so rotational delay is 6msec.

Transfer time: The time required to transfer one byte of data from track to read/write head or vice versa, is called Transfer time.

Disk access time: Disk access time is the total time it takes the computer to process a data request from the processor and then retrieve the required data from a storage device.

1.3 Record access

Indexing: An indexed file is a computer file with an index that allows easy random access to any record given its file key. The key must be such that it uniquely identifies a record. If more than one index is present the other ones are called alternate indexes. The indexes are created with the file and maintained by the system.

B-tree: A B-tree is a tree data structure that keeps data sorted and allows searches, insertions, and deletions in logarithmic amortized time. Unlike self-balancing

binary search trees, it is optimized for systems that read and write large blocks of data. It is most commonly used in database and file systems.

B+ tree: A B+ tree is an m-ary tree with a variable but often large number of children per node. A B+ tree consists of a root, internal nodes and leaves. A B+ tree can be viewed as a B-tree in which each node contains only keys (not key-value pairs), and to which an additional level is added at the bottom with linked leaves.

Hashing: Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value.

1.4 Document Processing

The definition of a document is quite restrictive for it specifies a substance (paper, in our case an electronic document), the method of making marks on it (writing or printing), and its use (furnishing information or evidence, legal or official). IMPROVEMENTS IN DOCUMENT PROCESSING - An economic appraisal of the problem of improving business data processing involves balancing the costs of making a change and using the new technique against the benefits derived. Important strides in mechanizing data processing are being made through the use of document reading devices. Character recognition machines are available from several manufacturers to read either one or perhaps a variety of type styles. Improvement in document processing should, in the rational world of economics, be determined by balancing:

- 1) The cost of operating the system utilizing the new technology versus the old.
- 2) Cost of transition from one to another.
- 3) The advantages derivable from the new technology.

Here we are using the document processing in such a way that we are searching for specific words in a certain electronic document. We can find and also replace the words we have found. It is done with the help of a concept called, Indexing. Indexing is explained in detail in the next subsection. Document processing is a very minor aspect in the File Structure field. Using concepts like indexing and hashing we can

expand the uses and application of File Structures.

1.5 Objective of the project

The main objective of the project is to index all the key words of the document in a separate file and perform operations like finding and replacing. The outcome of this project is to ease the user for finding and replacing words with a friendly, understandable GUI.

1.5.1 Problem Statement

This project has been created using Spyder, with a Windows platform. The project title is “DOCUMENT PROCESSING USING KEYWORDS” talks about how we process a text file and index the data and also use it to retrieve certain details in efficient way.

The purpose of this project is to reduce the time it takes to search keywords in a file by the help of computerized equipments and full-fledged computer software fulfilling their requirements, so that their valuable data/information can be retrieved faster with easily available and easy to work with.

It can assist the user to come up with a better document by looking at the keywords the system generates by processing the user’s document.

Scope of the Project

This project “Document Processing Using Keyword”, helps to search keywords in a text file.

In this process, the large text file is exposed to perform operations like search, modify and indexing where search helps to find keywords in a file, modify helps in altering the information and indexing helps to find the exact position of the words in the text file.

The purpose of this project is to reduce the time consumed to perform these operations by the help of computerized equipment and full-fledged computer software full filling their require.

CHAPTER 2

LITERATURE SURVEY

The project “DOCUMENT PROCESSING USING KEYWORDS” is designed to allow very fast full-text searches.

An inverted index consists of a list of all the unique words that appear in any document, and for each word, a list of

the documents in which it appears.

To process the document, the location of input is taken from the user. Main purpose of taking the location input is that the user can follow the operations for any file document. The operations done in this project are:

- **Indexing** (each word of the input text file is indexed and the word with the index is stored in a separate output file).
- **Searching** (the user inputs a word he wants to search in the document and that word is searched using the index and the lines containing that word are displayed).

INTRODUCTION TO PYTHON:

Python is a dynamic, interpreted (byte code-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

An excellent way to see how Python code works is to run the Python interpreter and type code right into it. If you ever have a question like, "What happens if I add an int to a list?" Just typing it into the Python interpreter is a fast and likely the best way to see what happens.

Like C++ and Java, Python is case sensitive so "a" and "A" are different

variables. The end of a line marks the end of a statement, so unlike C++ and Java, Python does not require a semicolon at the end of each statement. Comments begin with a '#' and extend to the end of the line.

Python source files use the ".py" extension and are called "modules." One unusual Python feature is that the whitespace indentation of a piece of code affects its meaning. A logical block of statements such as the ones that make up a function should all have the same indentation, set in from the indentation of their parent function or "if" or whatever. If one of the lines in a group has a different indentation, it is flagged as a syntax error.

CHAPTER 3

REQUIREMENT ANALYSIS AND DESIGN

2.1 Domain Understanding

The main object of the project is to index all the key words of the document in a separate file and perform operations like finding and replacing. The outcome of this project is to ease the user for finding and replacing words with a friendly, understandable GUI. To process the document, the location of input is taken from the user. Main purpose of taking the location input is that the user can follow the operations for any file document. The operations done in this project are:

- Indexing (each word of the input text file is indexed and the word with the index is stored in a separate output file)
- Searching (the user inputs a word he wants to search in the document and that word is searched using the index and the lines containing that word are displayed)

2.2 System Analysis

When the program is executed, it primarily asks the user to operate the operation presented that is to find.

When the user chooses the option to find, then it asks for the user to input the file in which he wants to “find” the word in. After the file is obtained, the entire file is indexed and the index values are maintained in a separate file.

After the completion of indexing, the user is prompted for the word he wants to search for. The user, if the word is present, is displayed with the number of times the word has been repeated and also in which line the word is found.

2.3 Software Requirements

- ☐ Operating System : UBUNTU / WINDOWS.
- ☐ Programming Language : Python.
- ☐ Tool Used : Spyder.

2.4 Hardware Requirements

- ☐ Processor : INTEL dual core and higher.
- ☐ MEMORY : 2GB and higher.
- ☐ HDD : Free space of 30 GB.
- ☐ Platform : Windows / Ubuntu.

CHAPTER 4

SYSTEM MODELLING

4.1 ARCHITECTURAL DESIGN:

Architectural design is a process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.”

Architectural design is a process for identifying the sub-systems making up a system and the framework for sub-system control and communication. The output of this design process is a description of the software architecture.

It represents the link between specification and design processes and is often carried out in parallel with some specification activities. It involves identifying major system components and their communications.

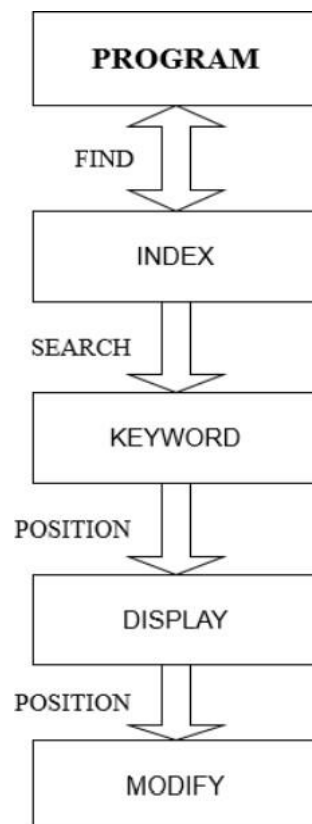


Fig 4.1: Architectural design of Document processing using keyword.

CHAPTER 5

IMPLEMENTATION

5.1 Indexing Algorithm

Step1: Open an input document that has to be processed.

Step2: Calculates the number of lines in the documents.

Step3: Open an empty document in write mode to store the indexes of the input document.

Step4: Reads all the lines in the document and split the words in it and returns the indexes of the respective words.

Step5: The words and the indexes in the output file. In the above algorithm firstly, it opens the document and it calculates the number of lines in the file and the document is read again and it opens another document in write mode and starts indexing using simple indexing algorithm. Each word in the document is splitted, and the splitted words are stored in an array and it writes it into a separate output file, which is later used in the program to perform search operation to find the specific word.

3.1 Searching Algorithm:

Step1: Open an input document that has to be processed.

Step2: Opens the index file and takes the user input for the word to be searched.

Step3: Search the given word in both input document and index file.

Step4: If the word is found, returns the entire line consisting the searched word with its index

Step5: Else it returns search is unsuccessful In the above algorithm firstly it opens

the input document and the index document, here we take the user input for the word to be searched using linear search algorithm. If the word is found it returns the word with its position or else it will return the word is not found.

5.2 Modifying Algorithm:

Step1: Takes user input whether the word need to be modified or not.

Step2: If yes, open the input document that has to be processed.

Step3: Prompts the user to enter the word to be replaced.

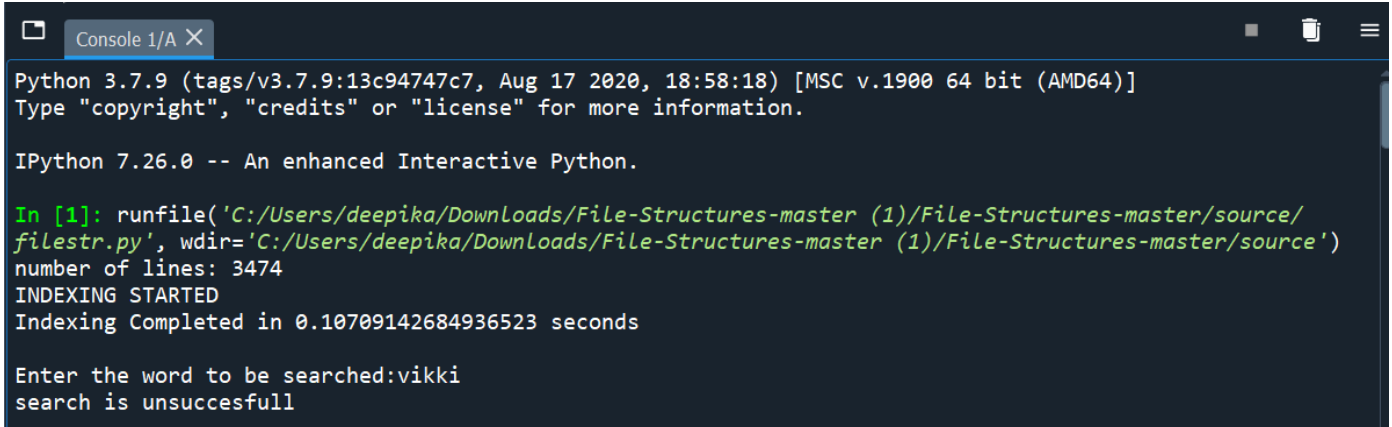
Step4: If the word exists in the document words will be replaced.

Step5: If no, it terminates and the file is closed. In the above algorithm it takes the user input whether the word to be modified or not. If the searched word is present in the document and if the user enters 'yes' its asks which word to be modified and asks which word to be replaced with. If the user enters 'no' it will terminate.

CHAPTER 6

SNAPSHOTS

Indexing



```

Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.26.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/deepika/Downloads/File-Structures-master (1)/File-Structures-master/source/
filestr.py', wdir='C:/Users/deepika/Downloads/File-Structures-master (1)/File-Structures-master/source')
number of lines: 3474
INDEXING STARTED
Indexing Completed in 0.10709142684936523 seconds

Enter the word to be searched:vikki
search is unsuccesfull

```

Fig 4.1

The above fig firstly returns the total number of lines in the input document. Starts indexing and stores the output file and calculates the time taken to complete indexing operation.



```

index2 - Notepad
File Edit Format View Help
SUDEEP 0
NAGARKARYOURE 6
THE 19
PASSWORD 22
TO 30
MY 32
LIFE 34
RANDOM 38
HOUSE 44
INDIAContentsA 49
Note 63
on 67
the 69
Author 72
PrologueA 78
Slightly 87
Insane 95
FriendshipMore 101
Than 115
Just 119
FriendsDhuaaIts 123
Hard 138
to 142
Let 144
GoLife 147
is 153
Not 155
Always 158
Fair 164
Your 168
Flaws 172
Are 177
Perfect 180
Were 187
We 191
Never 193
Meant 198
to 203
Be?Dreaming 205
of 216
YouYes, 218
I 225

```

Fig4.2

The above fig represents the output file with the indexes of each word.

Searching

```
In [1]: runfile('C:/Users/deepika/Downloads/File-Structures-master (1)/File-Structures-master/source/
filestr.py', wdir='C:/Users/deepika/Downloads/File-Structures-master (1)/File-Structures-master/source')
number of lines: 3474
INDEXING STARTED
Indexing Completed in 0.11345171928405762 seconds

Enter the word to be searched:Sudeep
```

Fig4.3

The above fig represents the searching operation where it takes the input from the user, which word to be searched.

```
Enter the word to be searched:sudeep
The line is here in index filewww.sudeepnagarkar.in 1002

www.sudeepnagarkar.in or get in touch with Sudeep via his:
Facebook fan page: facebook.com/sudeepnagarkar.official.fanpage
Facebook profile: facebook.com/sudeep.nagarkar
Twitter handle: sudeep_nagarkar
Email: contact@sudeepnagarkar.in

The line is here in index filefacebook.com/sudeepnagarkar.official.fanpageFacebook 1068

www.sudeepnagarkar.in or get in touch with Sudeep via his:
Facebook fan page: facebook.com/sudeepnagarkar.official.fanpage
Facebook profile: facebook.com/sudeep.nagarkar
Twitter handle: sudeep_nagarkar
Email: contact@sudeepnagarkar.in

The line is here in index filefacebook.com/sudeep.nagarkarTwitter 1128
```

Fig4.4

The above fig is the output where it displays indexes and the searched word exists.

```
The line is here in index filesudeep_nagarkarEmail: 1170
www.sudeepnagarkar.in or get in touch with Sudeep via his:
Facebook fan page: facebook.com/sudeepnagarkar.official.fanpage
Facebook profile: facebook.com/sudeep.nagarkar
Twitter handle: sudeep_nagarkar
Email: contact@sudeepnagarkar.in
The line is here in index filecontact@sudeepnagarkar.in... 1191
www.sudeepnagarkar.in or get in touch with Sudeep via his:
Facebook fan page: facebook.com/sudeepnagarkar.official.fanpage
Facebook profile: facebook.com/sudeep.nagarkar
Twitter handle: sudeep_nagarkar
Email: contact@sudeepnagarkar.in
Searching completed in: 0.063568115234375 seconds
```

Fig4.5

The above fig returns the time required to perform search operation.

```
You Gave Me a Hangover
The line is here in index fileHangover28th 131934
You Gave Me a Hangover
Searching completed in: 0.025003910064697266 seconds
Do you want to modify Y/N:y
Enter the word to be replaced:Hangover
Replace with:illness
Replaced Successfully in 0.00398707389831543 seconds
In [13]: runfile('C:/Users/deepika/Downloads/File-Structures-master (1)/File-Structures-master/source/
filestr.py', wdir='C:/Users/deepika/Downloads/File-Structures-master (1)/File-Structures-master/source')
number of lines: 3474
INDEXING STARTED
Indexing Completed in 0.13893342018127441 seconds
Enter the word to be searched:Hangover
search is unsuccessfull
```

Fig4.6

When the search word doesn't exist in the given file.

Modifying

```
number of lines: 3474
INDEXING STARTED
Indexing Completed in 0.12445998191833496 seconds

Enter the word to be searched:Hangover
The line is here in index fileHangoverA 258

You Gave Me a Hangover

The line is here in index fileHangover28th 131934

You Gave Me a Hangover

Searching completed in: 0.025003910064697266 seconds

Do you want to modify Y/N:y

Enter the word to be replaced:Hangover

Replace with:illness
Replaced Successfully in 0.00398707389831543 seconds
```

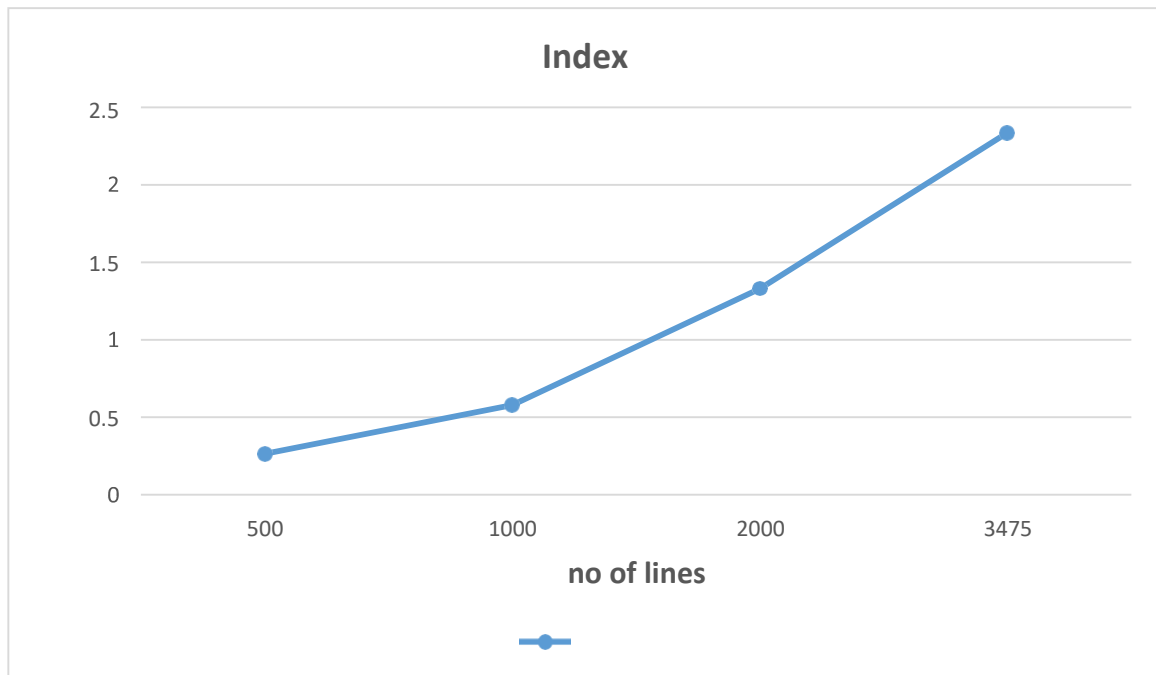
fig 4.7

The above fig represents modify operation where the user can replace any word in the file and it returns the time required to perform replace operation.

```
Do you want to modify Y/N:n
Done with the program
```

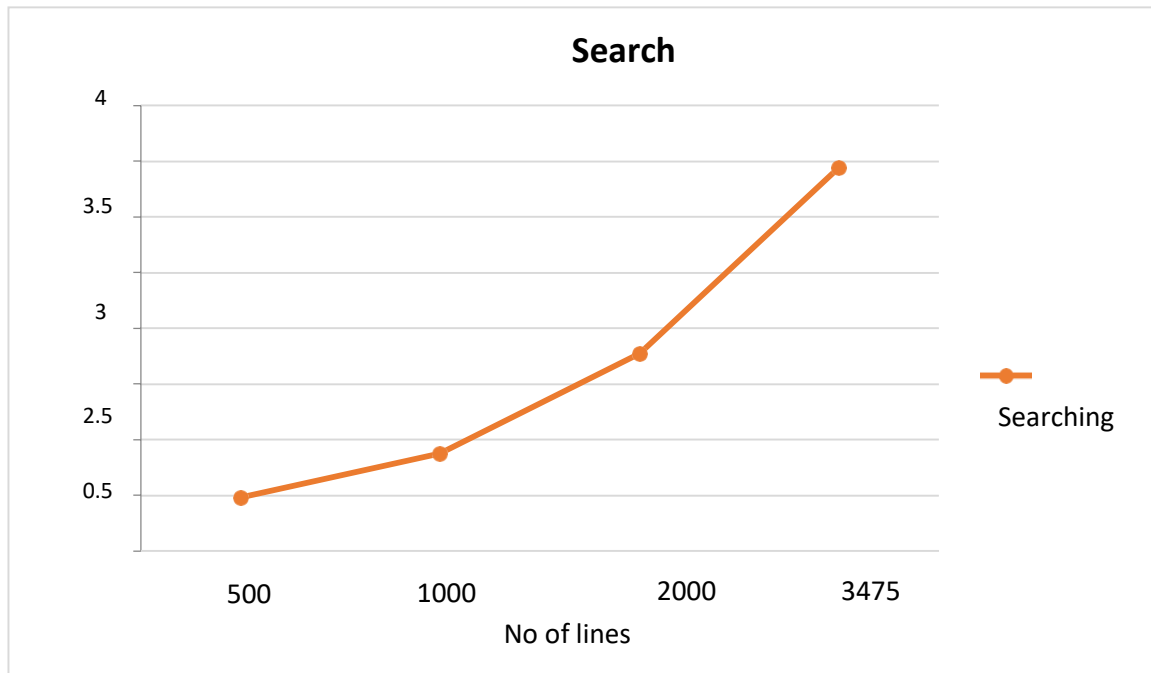
Fig4.8

When the search does not exist in the given input document.

Analysis:

Time taken for indexing

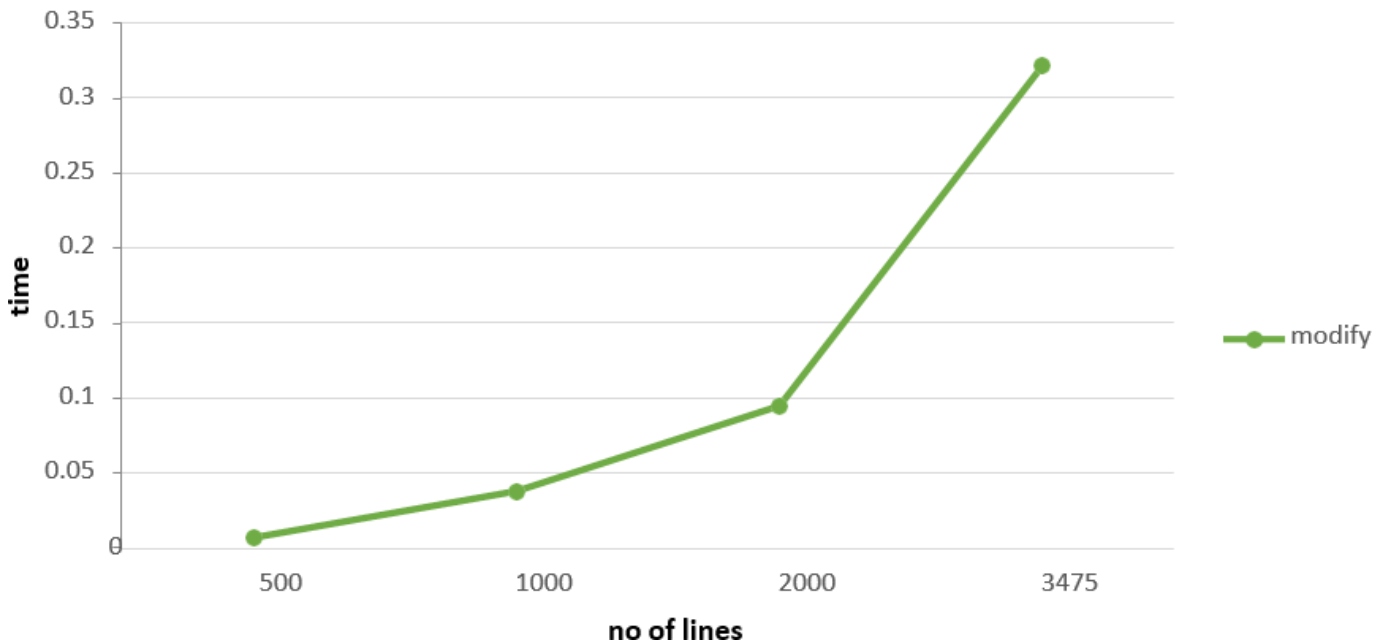
No of lines	Time taken for indexing(sec)
500	0.2636046
1000	0.578413
2000	1.32990017
3475	2.33436301



Time taken for searching

No of lines	Time taken for searching(sec)
500	0.479444
1000	0.87526
2000	1.7752
3475	3.443152

modify



Time taken for modifying

No of lines	Time taken for modifying(sec)
500	0.007168
1000	0.03788
2000	0.094628
3475	0.3217594

The above tables show the amount of time required to perform indexing, searching and modifying for different number of lines. Here in the Graph, we can clearly see that when the number of lines increase, even the time taken for indexing, searching and modifying increases. **Time** is represented in the Y-axis and **No. of lines** in the X-axis. As we increase the input, time drastically increases. Almost exponentially.

FUTURE WORK AND CONCLUSION

Future Work:

- We can add faster searching mechanisms.
- We can add a separate file for words which are searched often which will increase the probability of output in terms of speed.
- Showing where the words are in a separate window with the document.
- Creating a smoother GUI for better understanding and output display.

Conclusion

It was a huge learning experience as I took each aspect of the project seriously. My learning curve and also in coding has increased drastically with this mini project. Python has such vast domain features and anything can be accomplished. It was a very good learning experience in terms of time management and learning objectives. I hereby conclude this mini project “Document Processing Using Keywords” successfully with the trust of my senses and to best of my ability.

Python has such vast domain features and anything can be accomplished. It was a very good learning experience in terms of time management and learning objectives. We hereby conclude this mini project “Document Processing Using Keywords” successfully done to best of our ability.

REFERENCES

- [1] https://www.tutorialspoint.com/python/string_split.htm
- [2] <https://www.geeksforgeeks.org/python-strings/>
 - <https://app.diagrams.net/>
 - [Search · file structure mini project · GitHub](#)