

Visvesvaraya Technological University
Belagavi-590 018, Karnataka



A Mini Project Report on
**“Implementation of Hashing using Chaining Technique on sales record
dataset”**

**Mini Project Report submitted in partial fulfillment of the requirement
for the File Structures Lab [17ISL68]**

Bachelor of Engineering

In

Information Science and Engineering

Submitted by

Krupa P A [1JT17IS017]

Under the Guidance of

Mr.Vadiraja A

Asst. Professor

Dept. Of ISE



Department of Information Science and Engineering

Jyothy Institute of Technology

Tataguni, Bengaluru-560082

2020-21

Jyothy Institute of Technology
Tataguni, Bengaluru-560082
Department of Information Science and Engineering



CERTIFICATE

Certified that the mini project work entitled **“Implementation of Hashing using Chaining Technique on sales record dataset”** carried out by **Krupa P A [1JT17IS017]** bonfire student of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering in Information Science and Engineering** department of the **Visvesvaraya Technological University, Belagavi** during the year **2020-2021**. 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 Project work prescribed for the said Degree.

Mr .VadirajaA
Guide,Asst.Professor
Dept.Of ISE

Dr. HarshvardhanTiwari
Associate. Professor and HoD
Dept.Of ISE

ExternalVivaExaminer

Signature with Date:

- 1.
- 2.

ACKNOWLEDGEMENT

Firstly, we are very grateful to this esteemed institution “**Jyothy Institute of Technology**” for providing us an opportunity to complete our project.

I express my sincere thanks to our Principal **Dr. Gopalakrishna K** for providing us with adequate facilities to undertake this project.

I would like to thank **Dr. Harshvardhan Tiwari , Associate Prof. and Head** of Information Science and Engineering Department for providing his valuable support.

I would like to thank our guide **Mr.Vadiraja A , Asst. Prof.** for their keen interest and guidance in preparing this work.

Finally, I would thank all our friends who have helped me directly or indirectly in this project.

Krupa P A[1JT17IS017]

ABSTRACT

Hashing is a standard technique in the context of search for efficiently computing set similarities. The development of hashing provides a substantial improvement by storing hashed value. In this project, Hashing using Chaining collision technique for sales dataset is demonstrated using python language. Hashing is a data structure that is used to store a large number of data, which can be accessed in $O(1)$ time by operations such as search, insert, delete. Hashing is implemented in two steps: i) A key is converted into index by using a hash function. This index is used to store the original element, which falls into the hash table. ii) The element is stored in the hash table where it can be quickly retrieved using hashed key.

The situation where a newly inserted key maps to an already occupied slot in hash table is called *collision*. In this mini project separate chaining method is used to handle the collision. Here we can index and retrieve records in a file as it is faster to search that specific item using the shorter hashed key instead of using its original value. The direct location of the record on the disk can be calculated without using index structure.

TABLE OF CONTENTS

SL.NO	DESCRIPTION	PG NO.
	Chapter 1	
1.	Introduction	1
1.1	Introduction to File Structure	2
1.2	Introduction to Python	2
1.3	Introduction to Hashing	3
	Chapter 2	
2.	Implementation	5
2.1	Basic operations on Hashing	6
2.2	Algorithm	6
	Chapter 3	
3.	Hashing Analysis	7
3.1	Hashing Analysis	8
	Chapter 4	
4.	Results and Snapshots	9
4.1	Main function	10
4.2	Insertion of record	11
4.2.1	Insert Function	11
4.2.2	Hash Table	12
4.3	Deletion of record	13
4.3.1	Delete Function	13
4.3.2	Hash Table	14
4.4	Hash table generation	15
4.4.1	Hash Function	15
4.4.2	Hash Table	15
4.5	Searching a record from hash table	16
4.5.1	Search Function	16
4.5.2	Hash Table	17
	Conclusions	18
	References	19

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction to File Structure

- A disk's relatively slow access time and the enormous, nonvolatile capacity is the driving force behind FILE STRUCTURE design!!
- FS should give access to all the capacity without making the application spend a lot of time waiting for the disk.
- FS is a combination of representation for data in files and of operations for accessing the data.
 - It allows applications to read, write and modify data
 - Also finding the data
 - Or reading the data in a particular order
- Efficiency of FS design for a particular application is decided on,
 - Details of the representation of the data
 - Implementation of the operations
- A large variety in the types of data and in the needs of application makes FS design important.
- What is best for one situation may be terrible for other.

1.2 Introduction to Python

- Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.
- Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative, functional and procedural, and has a large and comprehensive standard library.
- Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation.

1.3 Introduction to Hashing

- Hashing is a data structure that is used to store a large number of data, which can be accessed in $O(1)$ time by operations such as search, insert, delete.
- Hashing is implemented in two steps:
 - A key is converted into index by using a hash function. This index is used to store the original element, which falls into the hash table.
 - The element is stored in the hash table where it can be quickly retrieved using hashed key.
- Using hashing we can index and retrieve records in a file as it is faster to search that specific item using the shorter hashed key instead of using its original value.
- The direct location of the record on the disk can be calculated without using index structure.

CHAPTER 2

IMPLEMENTATION

2.1 Basic operations on Hashing

In this section, we present the details of the operations of hashing:

- ✦ Inserting the record
- ✦ Deleting the record
- ✦ Build Hash Table
- ✦ Search

2.2 Algorithm

Step 1: CSV file of sales record is taken as input file.

Step 2: The first column of the CSV file is considered as key.

Step 3: Hash table is created and stored in a txt file.

Step 4: The key is then hashed into hash value.

Step 5: The hash value acts as index for the record in the hash table.

Step 6: The records are stored in its hash values in the hash table.

Step 7: If two or more keys get same hash value, then chaining method is used to avoid collision.

Step 8: In the above case, each record is linked to each other with same hash value and can be found using the same hash value.

Step 9: The insert function inserts the record into CSV file and then it is hashed like mentioned above. Therefore, the inserted record can be found both in CSV file and hash table.

Step 10: The delete function deletes the entire record by just giving its key value. The record is deleted both in CSV file and the hash table.

Step 11: The hash generation function generates the hash value for each key and stores the record in the hash table. It also helps us determine the time taken to hash the key values.

Step 12: The search function searches a record in hash table using the key value. As the hash value is the address of the record in the hash table, it directly goes to the location of the record and displays its value. This reduces the access to the disk while searching the required record.

2.3 Hash Algorithm used in this project

- Each character of the given key is converted into its ASCII value.
- Each ASCII value is added, and the result is divided by 100000 and the remainder is considered as the hash value.
- For example, if the key is OR0000123
 - The ASCII values of each character in the key is 79, 82, 48, 48, 48, 48, 49, 50, 51, respectively.
 - Now the ASCII values are added: $79+82+48+48+48+48+49+50+51 = 503$
 - The result is divided by 100000 and the remainder will be = 503
- For the above example, 503 is the hash value

CHAPTER 3

HASHING ANALYSIS

3.1 Hashing Analysis

The graph mentioned below gives the time analysis for no of records versus time taken for Hashing in seconds. In the first case, for 10,000 records, it takes time around 1.0189s for hashing. In the second case, for 20,000 records, it takes time around 2.95s for hashing. In the third case, for 30,000 records it takes time nearing to 6.1091s for hashing. For 40,000 records it takes time nearing to 11.05498s for hashing. In the third case, for 50,000 records it takes time nearing to 18.874s for hashing. In the third case, for 60,000 records it takes time nearing to 28.7118s for hashing. In the third case, for 70,000 records it takes time nearing to 39.77s for hashing. In the third case, for 80,000 records it takes time nearing to 70.34s for hashing. In the third case, for 90,000 records it takes time nearing to 65.3086s for hashing. In the last case, for 1,00,000 records it takes time nearing to 77.4794s for hashing.

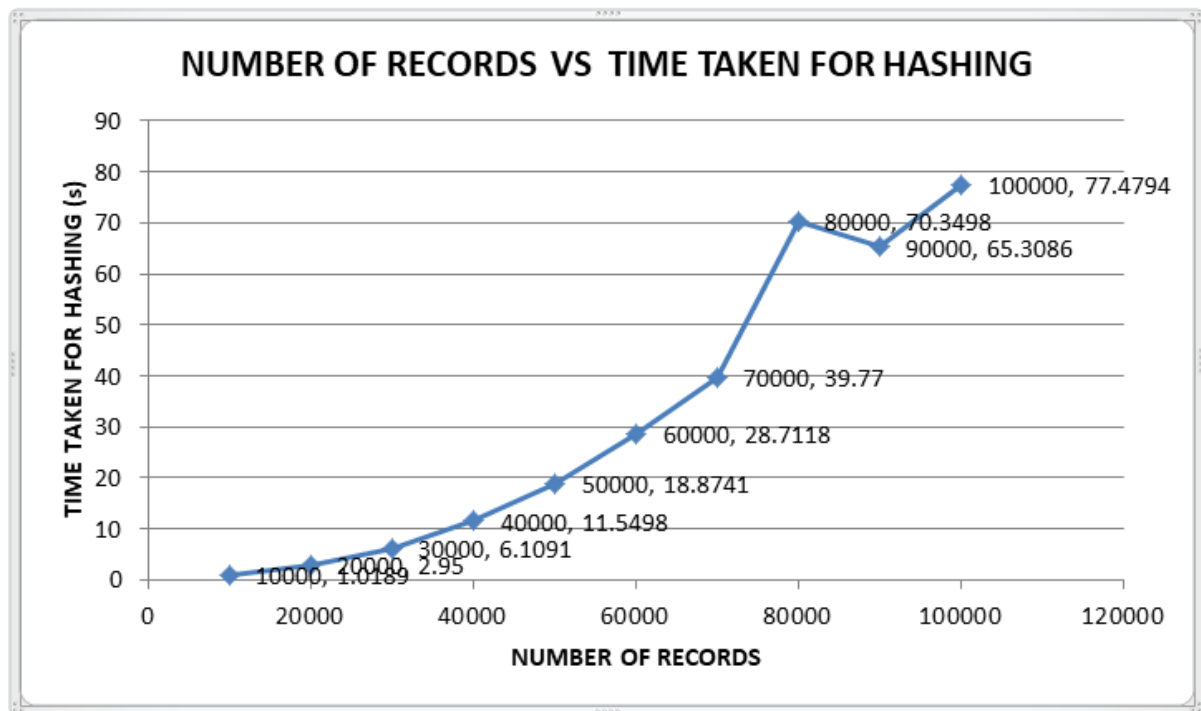
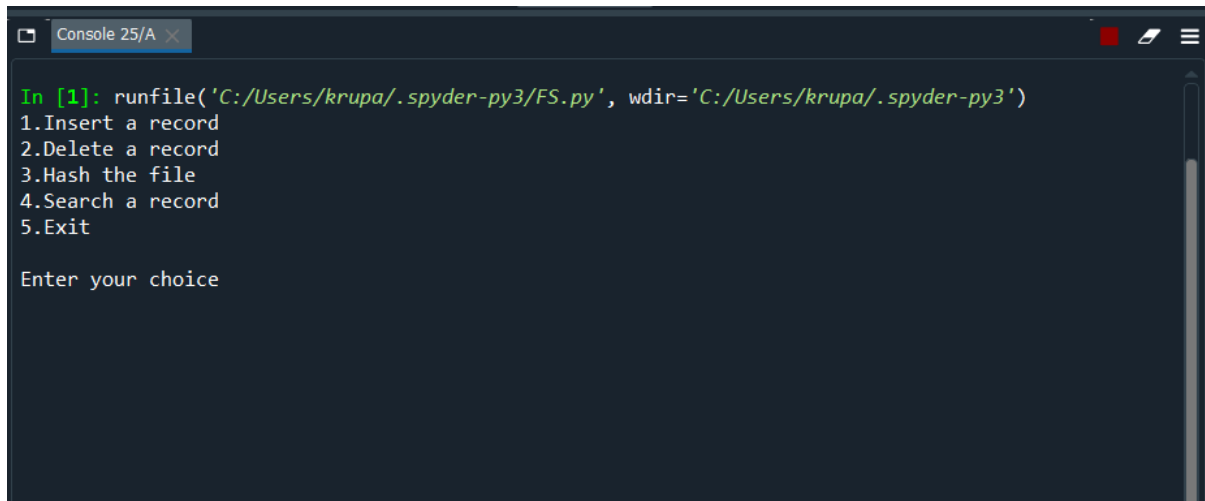


Fig 3.3.1 Time analysis for hashing

CHAPTER 4

RESULTS AND SCREENSHOTS

4.1 Main function

A screenshot of a Spyder IDE console window. The window title is 'Console 25/A'. The code in the console is as follows:

```
In [1]: runfile('C:/Users/krupa/.spyder-py3/FS.py', wdir='C:/Users/krupa/.spyder-py3')
1.Insert a record
2.Delete a record
3.Hash the file
4.Search a record
5.Exit

Enter your choice
```

Fig. 4.1 Main Function

This function provides the option for the operation that must be performed by the user.

The operations are:

1. Insert a record
2. Delete a record
3. Hash the file
4. Search a record
5. Exit

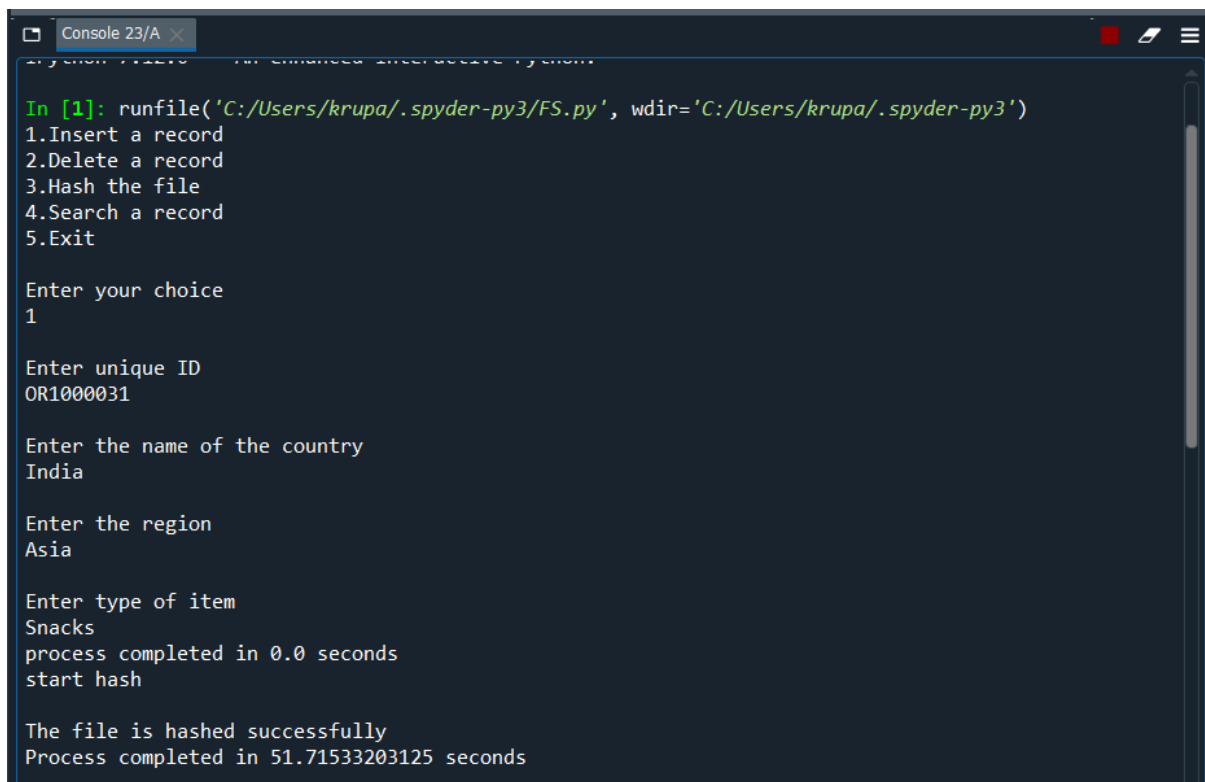
4.2 Insertion of the record

4.2.1 Insert function

Fig. 4.2.1 Insert Function

Using this function, we can enter the new record. In the above example:

Order ID: OR1000031



```
Console 23/A x
In [1]: runfile('C:/Users/krupa/.spyder-py3/FS.py', wdir='C:/Users/krupa/.spyder-py3')
1.Insert a record
2.Delete a record
3.Hash the file
4.Search a record
5.Exit

Enter your choice
1

Enter unique ID
OR1000031

Enter the name of the country
India

Enter the region
Asia

Enter type of item
Snacks
process completed in 0.0 seconds
start hash

The file is hashed successfully
Process completed in 51.71533203125 seconds
```

Country: India

Region: Asia

Item Type: Snacks

Is inserted and the time taken to insert is 0.0s

The time taken to hash is 51.71533203125

4. 2.2 Hash table

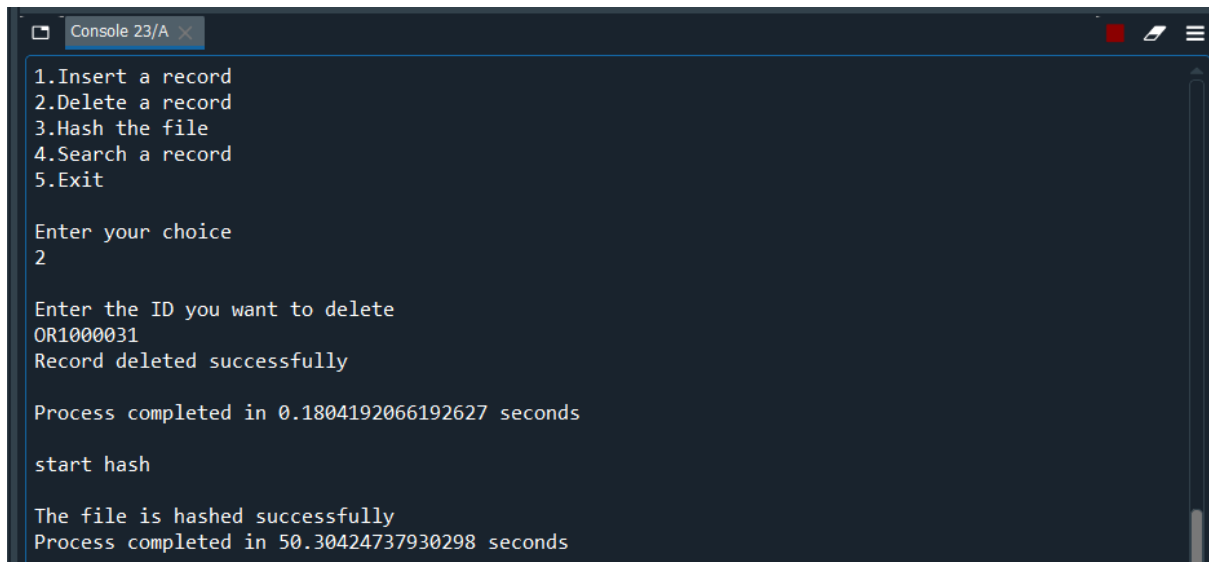


Fig. 4.2.2 Hash Table

From the above figure we can understand that the record is added in its hash value in the hash table.

4.3 Deletion of the record

4.3.1 Delete function



```
Console 23/A x
1.Insert a record
2.Delete a record
3.Hash the file
4.Search a record
5.Exit

Enter your choice
2

Enter the ID you want to delete
OR1000031
Record deleted successfully

Process completed in 0.1804192066192627 seconds

start hash

The file is hashed successfully
Process completed in 50.30424737930298 seconds
```

Fig. 4.3.1 Delete function

Using this function, we can delete the existing record just by giving its key. In the above example:

Order ID: OR1000031

Is given and the entire record of the given Order ID will be deleted and the time taken for the deletion is 0.1804192066192627 sec

The time taken to hash is 50.30424737930298 sec

4.3.2 Hash Table

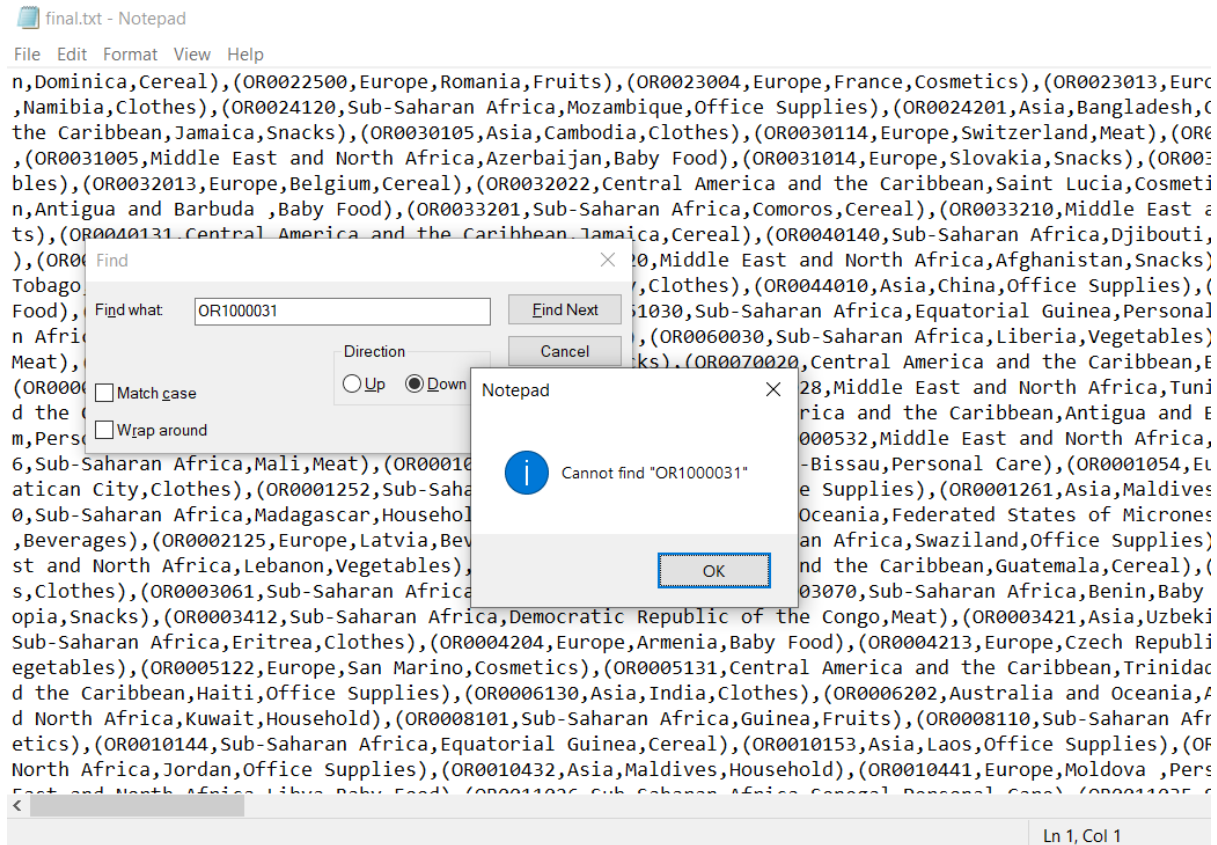
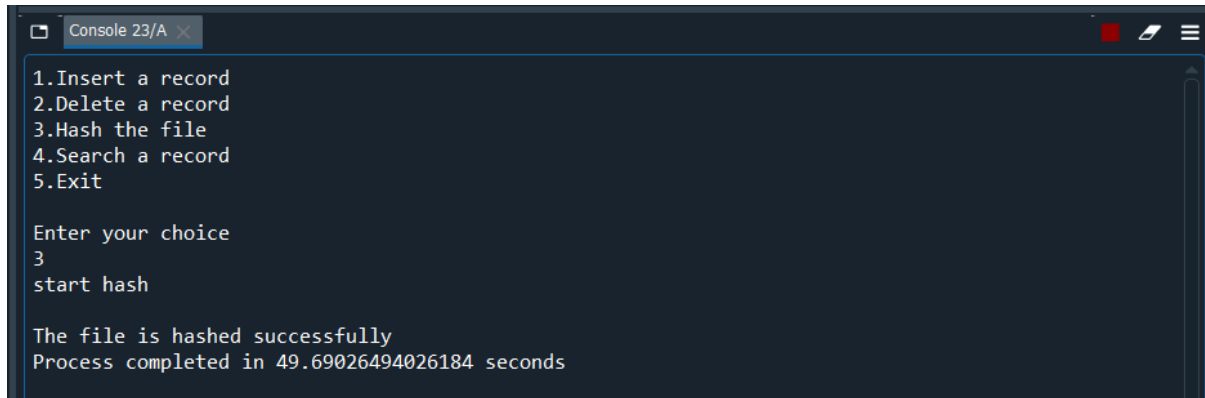


Fig. 4.3.2 Hash Table

From the above figure we can understand that the record with Order ID: OR1000031 is deleted from the hash table.

4.4 Hash table generation

4.4.1 Hash function



```

Console 23/A
1.Insert a record
2.Delete a record
3.Hash the file
4.Search a record
5.Exit

Enter your choice
3

start hash

The file is hashed successfully
Process completed in 49.69026494026184 seconds
  
```

Fig. 4.4.1 Hash Function

Using this function, all the records of the CSV file is hashed and all the records are stored in the hash table in its hash location.

Time taken to hash the entire 1lakh record is 49.69026494026184s.

4.4.2 Hash Table



```

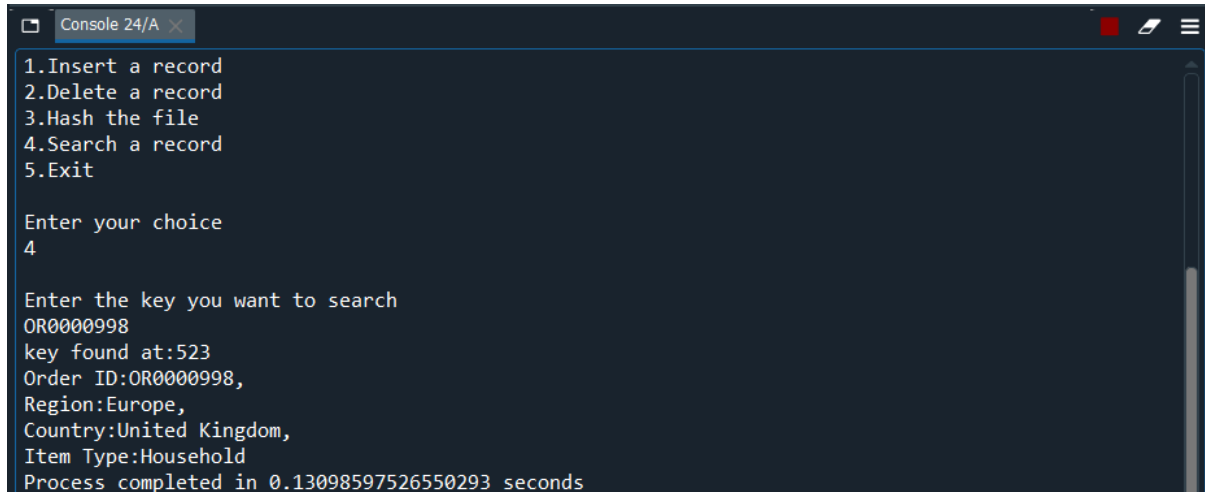
final.txt - Notepad
File Edit Format View Help
ice Supplies),(OR0039371,Sub-Saharan Africa,Sao Tome and Principe,Vegetables),(OR0039380,Central America and the Caribbean,Saint Lucia,Meat
Baby Food),(OR0039614,Sub-Saharan Africa,Gabon,Meat),(OR0039623,Europe,Russia,Vegetables),(OR0039632,Central America and the Caribbean,Nica
valu,Beverages),(OR0040388,Australia and Oceania,Vanuatu,Beverages),(OR0040397,Asia,India,Vegetables),(OR0040469,Sub-Saharan Africa,Guinea-
acks),(OR0040775,Australia and Oceania,Tonga,Cereal),(OR0040784,Australia and Oceania,Fiji,Fruits),(OR0040793,Asia,Thailand,Office Supplies
99,Asia,Uzbekistan,Office Supplies),(OR0041189,Asia,Japan,Clothes),(OR0041198,Central America and the Caribbean,Belize,Baby Food),(OR004127
,Asia,Indonesia,Vegetables),(OR0041648,Central America and the Caribbean,Barbados,Snacks),(OR0041657,Central America and the Caribbean,Domi
n Africa,Zambia,Clothes),(OR0041882,Europe,Poland,Snacks),(OR0041891,Europe,Portugal,Personal Care),(OR0041909,Europe,Liechtenstein,Meat),(
a and Oceania,Fiji,Clothes),(OR0042377,Europe,Czech Republic,Snacks),(OR0042386,North America,Greenland,Baby Food),(OR0042395,Asia,Nepal,Me
n Africa,Seychelles ,Personal Care),(OR0042683,Central America and the Caribbean,El Salvador,Cereal),(OR0042692,Sub-Saharan Africa,Botswana
(OR0042881,Europe,Luxembourg,Beverages),(OR0042890,Europe,Netherlands,Household),(OR0042908,Europe,Slovakia,Meat),(OR0042917,Europe,San Mar
Nevis ,Household),(OR0043286,North America,Canada,Vegetables),(OR0043295,Asia,Brunei,Cereal),(OR0043349,Europe,Malta,Clothes),(OR0043358,Eu
83,Asia,Laos,Fruits),(OR0043592,Middle East and North Africa,Israel,Cereal),(OR0043619,Asia,Laos,Household),(OR0043628,Sub-Saharan Africa,Si
enada,Snacks),(OR0043826,Europe,Cyprus,Cereal),(OR0043835,Sub-Saharan Africa,Guinea-Bissau,Snacks),(OR0043844,Sub-Saharan Africa,Sudan,Cosm
044177,Sub-Saharan Africa,Zimbabwe,Cereal),(OR0044186,Europe,Andorra,Household),(OR0044195,Sub-Saharan Africa,Cote d'Ivoire,Household),(OR0
a,Saudi Arabia,Cosmetics),(OR0044474,Europe,Czech Republic,Vegetables),(OR0044483,Central America and the Caribbean,Cuba,Vegetables),(OR004
2,Australia and Oceania,Papua New Guinea,Fruits),(OR0044681,North America,United States of America,Snacks),(OR0044690,Central America and th
Libya,Personal Care),(OR0044915,Asia,Brunei,Household),(OR0044924,Middle East and North Africa,Qatar,Meat),(OR0044933,Middle East and North
nacks),(OR0045275,Sub-Saharan Africa,Sao Tome and Principe,Fruits),(OR0045284,Middle East and North Africa,Tunisia ,Baby Food),(OR0045293,A
a,Cosmetics),(OR0045509,Sub-Saharan Africa,Cameroon,Office Supplies),(OR0045518,Central America and the Caribbean,The Bahamas,Cosmetics),(O
ages),(OR0045725,Europe,Italy,Clothes),(OR0045734,Middle East and North Africa,Somalia,Cereal),(OR0045743,Sub-Saharan Africa,Sao Tome and P
ca,Mali,Snacks),(OR0046067,Sub-Saharan Africa,Burkina Faso,Snacks),(OR0046076,Europe,Moldova ,Vegetables),(OR0046085,Middle East and North
Libya,Personal Care),(OR0046355,Asia,Turkmenistan,Fruits),(OR0046364,Europe,Ukraine,Cereal),(OR0046373,Asia,Japan,Personal Care),(OR0046382
at),(OR0046571,Central America and the Caribbean,Costa Rica,Household),(OR0046580,Asia,Maldives,Office Supplies),(OR0046607,Australia and O
ages),(OR0046850,Europe,Vatican City,Meat),(OR0046904,Asia,Kyrgyzstan,Meat),(OR0046913,Europe,France,Vegetables),(OR0046922,Central America
,Central America and the Caribbean,Dominican Republic,Clothes),(OR0047228,Sub-Saharan Africa,South Africa,Snacks),(OR0047237,North America,
s),(OR0047408,Sub-Saharan Africa,Eritrea,Fruits),(OR0047417,Middle East and North Africa,Somalia,Snacks),(OR0047426,Sub-Saharan Africa,Botsi
frica,Swaziland,Cereal),(OR0047651,Australia and Oceania,Vanuatu,Cereal),(OR0047660,Sub-Saharan Africa,Benin,Meat),(OR0047705,Sub-Saharan A
a,Office Supplies),(OR0048074,Europe,Monaco,Meat),(OR0048083,Middle East and North Africa,Jordan,Clothes),(OR0048092,Sub-Saharan Africa,Sou
Africa,Kyrgyzstan,Personal Care),(OR0048300,Middle East and North Africa,Yemen,Office Supplies),(OR0048309,Middle East and North Africa,Tunisi
  
```

Fig. 4.4.2 Hash Table

From the above the figure we can understand that the entire CSV file is hashed and stored in the hash table.

4.5 Searching a record from hash table

4.5.1 Searching function



```

Console 24/A x
1.Insert a record
2.Delete a record
3.Hash the file
4.Search a record
5.Exit

Enter your choice
4

Enter the key you want to search
OR0000998
key found at:523
Order ID:OR0000998,
Region:Europe,
Country:United Kingdom,
Item Type:Household
Process completed in 0.13098597526550293 seconds

```

Fig. 4.5.1 Searching Function

Using search function we can search a record from the hash table using its key. In the above example:

Order ID: OR0000998

Is the key given to search. The search function returns the value of the key quickly as it knows the location of the key in the hash table. Therefore, the values of the key such as:

Region: Europe

Country: United Kingdom

Item Type: Household

Is being displayed and the time taken to search this record is 36.8471071071201233ms

4.5.2 Hash Table

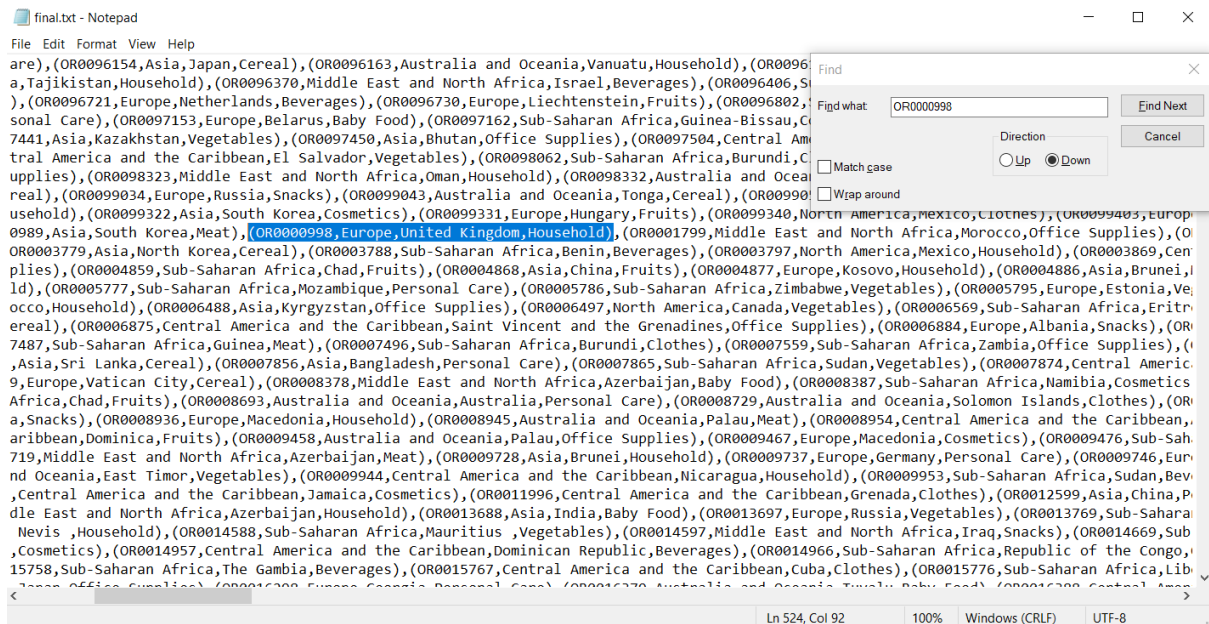


Fig. 4.5.2 Hash Table

From the above figure we can understand that the record displayed by the search function matches the record present in the hash table.

CONCLUSION

Hashing provides constant time search, insert, and delete operations on average.

This is why hashing is one of the most used data structure, example problems are, distinct elements, counting frequencies of items, finding duplicates, etc.

There are many other applications of hashing, including modern day cryptography hash functions. Some of these applications are listed below:

- Message Digest
- Password Verification
- Data Structures (Programming Languages)
- Compiler Operation
- Rabin-Karp Algorithm
- Linking File name and path together

In a large database, it is not possible to search all the indexes to obtain the required data. Hashing helps to find the direct location of a specific data record on the disk without using indexing.

From this project we can conclude that the hashing is providing constant time search, insert, and delete operations on average. And is more efficient than indexing.

REFERENCES

The information about Hashing was gathered by referring to the following sites:

- <https://www.youtube.com/watch?v=ea8BRGxGmlA>
- <https://www.youtube.com/watch?v=54iv1si4YCM>
- <https://stackoverflow.com/questions/62107924/how-can-i-convert-of-string-to-integer-value-in-a-list-in-python>
- <https://stackoverflow.com/questions/62108886/how-can-i-hash-hundred-thousand-records-taken-as-a-input-from-csv-file>
- <https://thispointer.com/python-read-a-csv-file-line-by-line-with-or-without-header/>
- <https://www.kite.com/python/answers/how-to-convert-all-items-in-a-list-to-floats-in-python>
- <https://www.edureka.co/community/30685/this-valueerror-invalid-literal-for-with-base-error-python>
- <https://www.youtube.com/watch?v=0Vl0iwkXrQ8>
- <https://www.geeksforgeeks.org/hashing-data-structure/>