

Complex Engineering Problem



Submitted by:

| | |
|---------------|-------------|
| Fariaa Faheem | 2019-EE-1 |
| Marwa Waseem | 2019-EE-7 |
| Hamza Akhtar | 2019-EE-12 |
| Filza Shahid | 2019-EE-151 |

Supervised by: Prof. Khalid Butt

Department of Electrical Engineering
University of Engineering and Technology Lahore

Complex Engineering Problem

Submitted to the faculty of the Electrical Engineering Department
of the University of Engineering and Technology Lahore
in partial fulfillment of the requirements for the Degree of

Bachelor of Science
in
Electrical Engineering.

Internal Examiner

External Examiner

Director
Undergraduate Studies

Department of Electrical Engineering
University of Engineering and Technology Lahore

Declaration

I declare that the work contained in this thesis is my own, except where explicitly stated otherwise. In addition this work has not been submitted to obtain another degree or professional qualification.

Signed: _____

Date: _____

Acknowledgments

Dedicated to

Contents

| | |
|------------------------------|-----|
| Acknowledgments | iii |
| List of Figures | vi |
| List of Tables | vii |
| 1 Problem Statement | 1 |
| 2 File Input | 2 |
| 3 Hash Table Implementation | 3 |
| 4 Array Implementation | 5 |
| 5 Linked List Implementation | 7 |
| 6 Tree Implementation | 8 |
| 7 Results | 10 |
| References | 11 |

List of Figures

| | | |
|-----|--|----|
| 3.1 | Results for hash implementation with data size 1000. | 3 |
| 3.2 | Results for hash implementation with data size 10000. | 3 |
| 3.3 | Results for hash implementation with data size 100000. | 3 |
| 3.4 | Results for hash implementation with data size 1000000. | 4 |
| 4.1 | Results for array implementation with data size 1000. | 5 |
| 4.2 | Results for array implementation with data size 10000. | 5 |
| 4.3 | Results for array implementation with data size 100000. | 6 |
| 4.4 | Results for array implementation with data size 1000000. | 6 |
| 6.1 | Results for tree implementation with data size 1000. | 8 |
| 6.2 | Results for tree implementation with data size 10000. | 8 |
| 6.3 | Results for tree implementation with data size 100000. | 9 |
| 6.4 | Results for tree implementation with data size 1000000. | 9 |
| 7.1 | Combined results for all data structures and operations. | 10 |

List of Tables

Chapter 1

Problem Statement

The main objectives of this Complex Engineering Problem are:

- To develop programs that store and manage the given data by using four different data structures, that are:
 1. Hash Table (Quadratic Probing)
 2. Array
 3. Linked List
 4. Binary Tree
- Implement the following operations on the given data:
 1. Insert all of the given data in a data structure.
 2. Print data of data structure in sorted order (traverse in sorted order) (numerically or alphabetically).
 3. Find records.
 4. Delete half of the data from the data structure.
- To measure execution time and memory consumption for each operation.
- To compare operations on different data structures depending on their execution time and memory consumption and conclude which data structure is the best for each operation.

Chapter 2

File Input

The workflow for file input routines is follows:

- File is opened in reading mode with **fopen()**
- Reading the data file using **fscanf()** from the first line. Ignore the first string read as it is needed. Next line contains the number of records read and it store as an integer. From the next line we have the data of employees.
- Creating a structure which has fields of "ID" (Integer), "Name" (String), "City" (String) and "Service" (String).
- Allocating memory for an array of "pointers to structures".
- Records are read from the file and are stored in the allocated structures and then pointers to these structures are linked to the array.
- File is closed after storing the data with **fclose**

So after these processes we have *an array of pointers to structures* containing the data of the files.

Chapter 3

Hash Table Implementation

```
Number of Records: 1000
Insert      Execution Time:    0.000054 s      Memory Consumption: 36280 bytes
Find        Execution Time:    0.000015 s      Memory Consumption: 36280 bytes
Sorted Traversal 101 Execution Time:    0.000120 s      Memory Consumption: 40280 bytes
Sorted Traversal      Execution Time:    0.008748 s      Memory Consumption: 40280 bytes
Delete          Execution Time:    0.000039 s      Memory Consumption: 40280 bytes

-----
Process exited after 0.02986 seconds with return value 0
Press any key to continue . . .
```

FIGURE 3.1: Results for hash implementation with data size 1000.

```
Number of Records: 10000
Insert      Execution Time:    0.000785 s      Memory Consumption: 360328 bytes
Find        Execution Time:    0.000317 s      Memory Consumption: 360328 bytes
Sorted Traversal 101 Execution Time:    0.002094 s      Memory Consumption: 400328 bytes
Sorted Traversal      Execution Time:    0.864026 s      Memory Consumption: 400328 bytes
Delete          Execution Time:    0.000342 s      Memory Consumption: 400328 bytes

-----
Process exited after 0.8982 seconds with return value 0
Press any key to continue . . .
```

FIGURE 3.2: Results for hash implementation with data size 10000.

```
Number of Records: 100000
Insert      Execution Time:    0.006306 s      Memory Consumption: 3600040 bytes
Find        Execution Time:    0.002778 s      Memory Consumption: 3600040 bytes
Sorted Traversal 101 Execution Time:    0.018601 s      Memory Consumption: 4000040 bytes
Sorted Traversal      Execution Time:    90.938276 s      Memory Consumption: 4000040 bytes
Delete          Execution Time:    0.003380 s      Memory Consumption: 4000040 bytes

-----
Process exited after 91.73 seconds with return value 0
Press any key to continue . . .
```

FIGURE 3.3: Results for hash implementation with data size 100000.

```
Number of Records: 1000000
Insert      Execution Time:    0.090509 s      Memory Consumption:  36000184 bytes
Find       Execution Time:    0.038666 s      Memory Consumption:  36000184 bytes
Delete     Execution Time:    0.046859 s      Memory Consumption:  36000184 bytes
-----
Process exited after 1.562 seconds with return value 0
Press any key to continue . . .
```

FIGURE 3.4: Results for hash implementation with data size 1000000.

Chapter 4

Array Implementation

```
Number of Records: 1000
Insert          Execution Time: 0.000036 s      Memory Usage: 24016 bytes
Find           Execution Time: 0.000594 s      Memory Usage: 24016 bytes
Sorted Traversal Execution Time: 0.000103 s      Memory Usage: 24016 bytes
Delete         Execution Time: 0.000841 s      Memory Usage: 24016 bytes

-----
Process exited after 0.01723 seconds with return value 0
Press any key to continue . . .
```

FIGURE 4.1: Results for array implementation with data size 1000.

```
Number of Records: 10000
Insert          Execution Time: 0.000146 s      Memory Usage: 240016 bytes
Find           Execution Time: 0.064914 s      Memory Usage: 240016 bytes
Sorted Traversal Execution Time: 0.001254 s      Memory Usage: 240016 bytes
Delete         Execution Time: 0.057418 s      Memory Usage: 240016 bytes

-----
Process exited after 0.1514 seconds with return value 0
Press any key to continue . . .
```

FIGURE 4.2: Results for array implementation with data size 10000.

```
Number of Records: 100000
Insert      Execution Time: 0.001644 s      Memory Usage: 2400016 bytes
Find       Execution Time: 5.140638 s      Memory Usage: 2400016 bytes
Sorted Traversal Execution Time: 0.015411 s  Memory Usage: 2400016 bytes
Delete     Execution Time: 5.141023 s      Memory Usage: 2400016 bytes

-----
Process exited after 10.53 seconds with return value 0
Press any key to continue . . .
```

FIGURE 4.3: Results for array implementation with data size 100000.

```
Number of Records: 1000000
Insert      Execution Time: 0.016733 s      Memory Usage: 24000016 bytes
Find       Execution Time: 706.543014 s     Memory Usage: 24000016 bytes
Delete     Execution Time: 770.063951 s     Memory Usage: 24000016 bytes

-----
Process exited after 1480 seconds with return value 0
Press any key to continue . . .
```

FIGURE 4.4: Results for array implementation with data size 1000000.

Chapter 5

Linked List Implementation

Chapter 6

Tree Implementation

```
Number of Records: 1000
Insert      Execution Time:    0.000338 s      Memory Consumption:  40000 bytes
Sorted Traversal Execution Time: 0.000013 s      Memory Consumption:  40000 bytes
Find       Execution Time:    0.000055 s      Memory Consumption:  40000 bytes
Delete     Execution Time:    0.000080 s      Memory Consumption:  20000 bytes

-----
Process exited after 0.02053 seconds with return value 0
Press any key to continue . . .
```

FIGURE 6.1: Results for tree implementation with data size 1000.

```
Number of Records: 10000
Insert      Execution Time:    0.003853 s      Memory Consumption:  400000 bytes
Sorted Traversal Execution Time: 0.000120 s      Memory Consumption:  400000 bytes
Find       Execution Time:    0.000838 s      Memory Consumption:  400000 bytes
Delete     Execution Time:    0.001143 s      Memory Consumption:  200000 bytes

-----
Process exited after 0.03379 seconds with return value 0
Press any key to continue . . .
```

FIGURE 6.2: Results for tree implementation with data size 10000.


```
Number of Records: 100000
Insert           Execution Time:    0.072999 s      Memory Consumption:  4000000 bytes
Sorted Traversal Execution Time:    0.002779 s      Memory Consumption:  4000000 bytes
Find            Execution Time:    0.017485 s      Memory Consumption:  4000000 bytes
Delete          Execution Time:    0.026487 s      Memory Consumption:  2000000 bytes

-----
Process exited after 0.2511 seconds with return value 0
Press any key to continue . . .
```

FIGURE 6.3: Results for tree implementation with data size 100000.

```
Number of Records: 1000000
Insert           Execution Time:    1.252477 s      Memory Consumption:  40000000 bytes
Sorted Traversal Execution Time:    0.050533 s      Memory Consumption:  40000000 bytes
Find            Execution Time:    0.344922 s      Memory Consumption:  40000000 bytes
Delete          Execution Time:    0.467762 s      Memory Consumption:  20000000 bytes

-----
Process exited after 2.996 seconds with return value 0
Press any key to continue . . .
```

FIGURE 6.4: Results for tree implementation with data size 1000000.

Chapter 7

Results

| No. of Records | Data Structure | Execution Time (s) | | | | | Memory Consumption (bytes) | | | | |
|----------------|----------------|--------------------|------------|----------|-----------|------------|----------------------------|----------|----------|---------|----------|
| | | Insert | Find | Sorted | | Delete | Insert | Find | Sorted | | Delete |
| 1000 | Hash Table | 0.000054 | 0.000015 | 0.000120 | 0.008748 | 0.000039 | 36280 | 36280 | 40280 | 36280 | 36280 |
| | Array | 0.000036 | 0.000594 | 0.000103 | | 0.000841 | 24016 | 24016 | 24016 | | 24016 |
| | Linked List | | | | | | | | | | |
| | Tree | 0.000338 | 0.000055 | 0.000013 | | 0.000080 | 40000 | 40000 | 40000 | | 20000 |
| 10000 | Hash Table | 0.000785 | 0.000317 | 0.002094 | 0.864086 | 0.000342 | 360328 | 360328 | 400328 | 360328 | 360328 |
| | Array | 0.000146 | 0.064914 | 0.001254 | | 0.057418 | 240016 | 240016 | 240016 | | 240016 |
| | Linked List | | | | | | | | | | |
| | Tree | 0.003853 | 0.000838 | 0.000120 | | 0.001143 | 400000 | 400000 | 400000 | | 200000 |
| 100000 | Hash Table | 0.006306 | 0.002778 | 0.018601 | 90.938276 | 0.003380 | 3600040 | 3600040 | 4000040 | 3600040 | 3600040 |
| | Array | 0.001644 | 5.140638 | 0.015411 | | 5.141023 | 2400016 | 2400016 | 2400016 | | 2400016 |
| | Linked List | | | | | | | | | | |
| | Tree | 0.072999 | 0.017485 | 0.002779 | | 0.026487 | 4000000 | 4000000 | 4000000 | | 2000000 |
| 1000000 | Hash Table | 0.090509 | 0.038666 | - | - | 0.046859 | 36000184 | 36000184 | - | - | 36000184 |
| | Array | 0.016733 | 706.543014 | - | | 770.063951 | 24000016 | 24000016 | - | | 24000016 |
| | Linked List | | | - | | | | | - | | |
| | Tree | 1.252477 | 0.344922 | 0.050533 | | 0.467762 | 40000000 | 40000000 | 40000000 | | 20000000 |

FIGURE 7.1: Combined results for all data structures and operations.

References

[1]