Contact Management System

Adil Chaka, Joy Son
Dept. of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, USA
Email: achaka@stevens.edu, json2@stevens.edu

***Abstract*—**In this study, we present a Contact Management System (CMS) tailored to enhance the efficiency and responsiveness of managing personal and professional contacts. Leveraging data structures such as arrays, linked lists, and hash tables, the system is designed to optimize operations including search, addition, deletion, and updating of contacts. Our findings reveal that each data structure uniquely impacts the system's performance, with hash tables providing the most significant improvements in search operations. Through comprehensive performance tests and analysis, this paper illustrates the practical benefits and potential applications of the CMS in real-world scenarios, offering a scalable solution to contact management challenges.

I. Introduction

Contact management systems are pivotal in both personal and business environments, aiding in the organization and retrieval of contact information swiftly. With the burgeoning amount of data handled daily, traditional contact management methods have become insufficient, prompting the need for more sophisticated solutions. This paper introduces a CMS designed to address these challenges through the effective use of fundamental data structures: arrays, linked lists, and hash tables.

This project was initiated to explore efficient data handling techniques that could significantly reduce search and manipulation times. The ensuing sections detail the system's architecture, the rationale behind the chosen data structures, implementation specifics, a thorough performance evaluation, and our conclusive insights.

II. Design

Overall Architecture

The CMS architecture consists of three primary components:

1.) **User Interface (UI):** A graphical user interface that allows users to interact with the CMS to perform operations such as add, delete, update, and search contacts.
2.) **Logic Layer:** Implements the business logic of the CMS, interfacing between the UI and data storage. It processes user requests, executes appropriate data operations, and returns results.
3.) **Data Storage:** Utilizes arrays, linked lists, and hash tables to store contact information efficiently.

Data Structures

- **Arrays:** Used for maintaining a sorted list of contacts, optimizing the listing operation with direct indexing.
- **Linked Lists:** Facilitate dynamic data management where contacts can be added or removed without reallocating the entire data structure.
- **Hash Tables:** Provide an efficient way of indexing and retrieving data, significantly reducing the time complexity for search operations from O(n) to an average-case time complexity of O(1).

User Interface Design

The UI provides a simple and intuitive interface, featuring forms for entering new contact data, options to update or remove existing contacts, and a search bar for quick retrieval. UI mockups (not

included here) delineate the layout and interaction flow, ensuring user-friendly navigation.

III. Implementation

Development Environment

The Contact Management System was implemented in Python due to its simplicity, readability, and the vast array of libraries that support rapid development and testing. We utilized Python 3.8 and leveraged several modules such as collections for data structures like hash tables (dictionaries) and lists for linked lists. Development was facilitated by the PyCharm IDE, and version control was managed with Git, hosted on GitHub to streamline collaboration and revision tracking.

Key Function Implementations

Below are detailed explanations and code snippets of critical functions within the system:

Adding Contacts

Using a hash table implemented through Python's dictionary, we can efficiently manage additions as follows:

```
def add_contact(contact_book, name, phone_number):
    """Add a new contact to the hash table."""
    if name not in contact_book:
        contact_book[name] = phone_number
        print(f"Contact {name} added successfully.")
    else:
        print("Contact already exists.")
```

This function checks if the contact already exists to prevent duplicates and adds a new contact if not present.

Searching for Contacts

Here's how a contact can be searched in the hash table, providing a quick retrieval based on the contact's name:

```
def search_contact(contact_book, name):
    """Search for a contact by name."""
    try:
        print(f"Contact: {name}, Phone Number: {contact_book[name]}")
    except KeyError:
        print("Contact not found.")
```

This function utilizes Python's exception handling to manage cases where the contact does not exist, thereby maintaining efficient search capabilities.

Deleting Contacts

Contacts can be removed from the hash table using the following method:

```
def delete_contact(contact_book, name):
    """Delete a contact from the hash table."""
    if name in contact_book:
        del contact_book[name]
        print(f"Contact {name} removed successfully.")
    else:
        print("Contact does not exist.")
```

This method checks for the contact's existence and removes it if found, providing feedback on the action's success or failure.

Updating Contacts

Updating contact information is crucial and can be handled as follows:

```
def update_contact(contact_book, name, new_phone_number):
    """Update an existing contact's phone number."""
    if name in contact_book:
```

```
        contact_book[name] = new_phone_number
        print(f"Contact {name} updated
successfully.")
    else:
        print("Contact does not exist.")
```

This function ensures that updates are only made to existing entries, thus safeguarding the integrity of the data.

IV. Analysis and Discussion

V. Conclusion

VI. References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," MIT Press, 3rd ed., 2009.

[2] A. B. Downey, "Think Python: How to Think Like a Computer Scientist," O'Reilly Media, 2nd ed., 2016.

[3] M. L. Hetland, "Python Algorithms: Mastering Basic Algorithms in the Python Language," Apress, 2nd ed., 2017.

[4] M. Lutz, "Learning Python," O'Reilly Media, 5th ed., 2013.

[5] Python Software Foundation, Python Documentation, Available online: https://docs.python.org/3/ [Accessed on 4/16/2024].

[6] D. Saha, "Data Structures and Algorithms with Python," Springer, 2018.

[7] R. Sedgewick and K. Wayne, "Algorithms," Addison-Wesley Professional, 4th ed., 2011.