DSA GROUP PROJECT: Phonebook Application

1. Modules

- **Insert contact:** Adds a new contact to the phonebook.
- **Search contact:** Searches for a contact by name or number.
- **Display contacts:** Displays all contacts in the phonebook.
- **Delete contact:** Deletes a contact by name or number.
- **Update contact:** Updates an existing contact's details.
- **Sort contacts:** Sorts contacts for optimized search.
- **Analyze search efficiency:** Analyzes the time complexity of the search algorithm.

2. Functions

Insert contact (phonebook, name, number):

- Input: phonebook (list or array), name (string), number (string)
- Functionality: Adds a new contact (name and number) to the phonebook.

Search contact (phonebook, search term):

- Input: phonebook (list or array), search term (string)
- Functionality: Searches for the contact using a linear search or binary search (if sorted).

Display contacts (phonebook):

- Input: phonebook (list or array)
- Functionality: Displays all contacts stored in the phonebook in a user-friendly format.

Delete contact (phonebook, name):

- Input: phonebook (list or array), name (string)
- Functionality: Removes the contact that matches the given name.

Update contact (phonebook, name, new number):

- Input: phonebook (list or array), name (string), new number (string)
- Functionality: Updates the number for a contact identified by the name.

Sort contacts (phonebook):

- Input: phonebook (list or array)
- Functionality: Sorts the contacts alphabetically by name to optimize searching.

Analyze search efficiency (phonebook, search term):

- Input: phonebook (list or array), search term (string)
- Functionality: Analyzes and prints the time complexity of search for linear vs. binary search.

3. Pseudocode

1. Define Contact Class:

- **Attributes:**
  - name, phone, email (optional)
- **Methods:**
  - __init__(self, name, phone, email=""): Initializes attributes.
  - __str__(self): Returns a formatted string with the contact's details.

2. Define Phonebook Class:

- **Attribute:**
  - contacts (a list to store Contact objects)
- **Methods:**
  - **insert_contact(self, name, phone, email=""):**
    - Create a new Contact instance.
    - Add it to the contacts list.
  - **search_contact(self, query):**
    - Search for contacts where the name or phone matches the query.
    - Return matching contacts.
  - **delete_contact(self, query):**
    - Remove contacts from the list if their name or phone matches the query**.**
  - **update_contact(self, old_query, new_name, new_phone, new_email=""):**
    - Find a contact matching old_query.
    - Update the name, phone, and email.

        - Return True if successful, otherwise False.
    - **sort_contacts(self):**
        - Sort contacts list by the name attribute.
    - **display_contacts(self):**
        - Return a formatted string of all contacts**.**

3. Define PhonebookApp Class (GUI Application):

- **Attributes:**
    - phonebook: An instance of Phonebook.
    - root: The Tkinter root window.
- **Constructor (__init__(self, root)):**
    - Initialize the GUI window with:
        - Title, size, background color, and styles for buttons and labels.
    - Create buttons for different functionalities (Insert, Search, Display, Delete, Update, Sort).
    - Add a footer.
- **Methods for Button Actions:**
    - **insert_contact(self):**
        - Prompt the user for name, phone, and email (optional) using a dialog box.
        - If name and phone are provided, insert the contact into phonebook.
        - Show a success or error message.
    - **search_contact(self):**
        - Prompt the user for a query (name or phone).
        - Search the phonebook and display results.
        - Show appropriate messages based on the search results.
    - **display_contacts(self):**
        - Display all contacts if available.
        - Show an appropriate message if no contacts exist.

- **delete_contact(self):**
  - Prompt the user for a query (name or phone).
  - Delete matching contacts from phonebook.
  - Show success or error messages.
- **update_contact(self):**
  - Prompt the user for old_query (name or phone) to find the contact.
  - If found, prompt for new details (name, phone, email).
  - Update the contact and display success or failure messages.
- **sort_contacts(self):**
  - Sort contacts by name and show a confirmation message.

4. Main Application Entry Point:

- Initialize the Tkinter root window.
- Create an instance of PhonebookApp.
- Run the main event loop (root.mainloop()).

Analyze search efficiency:

Function analyze_search_efficiency (phonebook, search_term):

If phonebook is sorted:

Perform binary search and calculate time complexity

Else:

Perform linear search and calculate time complexity

End

Print time complexity for the search

End