Data Structures : Algorithms and Applications(Lab)

EXPERIMENT No: 12

AIM: Implementation of Binary Search in real life application

THEORY:

Binary search is an efficient algorithm for finding a target value within a sorted array. It operates by repeatedly dividing the search interval in half, thus reducing the time complexity to O(logn)O(\log n)O(logn). The basic steps of binary search are:

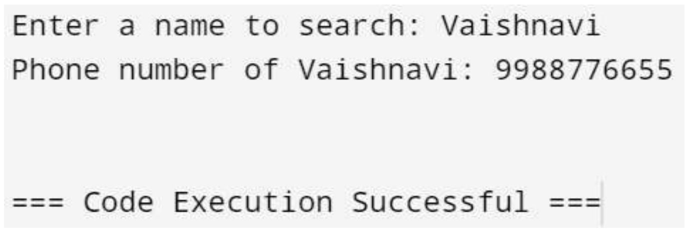
1. Initialization: Start with two pointers, left and right, representing the boundaries of the search interval.
2. Middle Element: Calculate the middle index of the current search interval.
3. Comparison: Compare the target value to the middle element:
   - If they are equal, the search is successful.
   - If the target value is less than the middle element, narrow the search to the left half.
   - If the target value is greater, narrow the search to the right half.
4. Repeat: Continue the process until the target value is found or the interval is empty.

In the context of a phone book, binary search allows for quick lookups of phone numbers based on names. The names are stored in a sorted array along with their corresponding phone numbers, allowing users to efficiently retrieve contact information without needing to scan each entry sequentially.

CODE:
```
#include <stdio.h>
#include <string.h>

#define MAX 100

// Structure to store phone book entries
struct PhoneBookEntry {
  char name[50];
  char phoneNumber[15];
};

// Function for binary search
int binarySearch(struct PhoneBookEntry phoneBook[], int n, const char* target) {
  int left = 0, right = n - 1;

  while (left <= right) {
    int mid = left + (right - left) / 2;
```

```
        // Compare middle entry with target
        int cmp = strcmp(phoneBook[mid].name, target);
        if (cmp == 0) {
            return mid; // Found
        }
        if (cmp < 0) {
            left = mid + 1; // Search right
        } else {
            right = mid - 1; // Search left
        }
    }
    return -1; // Not found
}

// Driver code
int main() {
    struct PhoneBookEntry phoneBook[MAX] = {
        {"Gauri", "9146898345"},
        {"Neha", "9123456789"},
        {"Supriya", "9876543210"},
        {"Vaishnavi", "9988776655"}
    };

    int n = 4; // Number of entries in the phone book
    char target[50];

    // Sorting is assumed to be done before searching
    printf("Enter a name to search: ");
    scanf("%s", target);

    int result = binarySearch(phoneBook, n, target);
    if (result != -1) {
        printf("Phone number of %s: %s\n", target, phoneBook[result].phoneNumber);
    } else {
        printf("Name not found in the phone book.\n");
    }

    return 0;
}
```

OUTPUT:



```
Enter a name to search: Vaishnavi
Phone number of Vaishnavi: 9988776655


=== Code Execution Successful ===
```

CONCLUSION:
Binary search efficiently retrieves phone numbers from a sorted phone book by quickly locating names, demonstrating its effectiveness in real-life applications. This method significantly reduces search time compared to linear search, enhancing user experience.