

**Symbiosis Institute of Technology**

**Faculty of Engineering**

**CSE- Academic Year 2024-25**

**Data Structures – Lab Batch 2023-27**

|  |  |
| --- | --- |
| **Lab Assignment No:- 1** | |
|  | |
| **Name of Student** | Pankhuri Varshney |
| **PRN No.** | 23070122160 |
| **Batch** | 2023-27 |
| **Class** | CS-B2 |
| **Academic Year & Semester** | 2024-25  Semester 3 |
| **Date of Performance** | 23rd July, 2024 |
|  | |
| **Title of Assignment:** | A. Implement following searching algorithm: Linear search with multiple occurrences  B. Implement following searching algorithms in menu:  1. Binary search with iteration  2. Binary search with recursion |
| **Theory Questions:** | 1. Prepare table for following 5 different searching algorithms for their best case, average case and worst case time complexities. 2. Apply binary search on the input data set. Show all steps. 3. Compare linear search and binary search 4. Why is the complexity of Binary search O(log n)? |
| **Source Code/Algorithm/Flow Chart:** | **A. Implement following searching algorithm: Linear search with multiple occurrences**  #include <stdio.h>  void linearSearch(int arr[], int n, int key);  int rec\_linearSearch(int arr[], int n, int key);  int main(){  printf("Enter size of an array: ");  int n, key;  scanf("%d", &n);  int arr[n];  printf("Enter the elements\n");  for(int i=0; i<n; i++)  scanf("%d", &arr[i]);  printf("Enter the key value to be searched: ");  scanf("%d", &key);  linearSearch(arr, n, key);  printf("USING RECURSION: \n");  if(rec\_linearSearch(arr, n, key)==-1)  printf("KEY NOT FOUND\n");  else  printf("KEY FOUND AT INDEX: %d", rec\_linearSearch(arr, n, key));  }  void linearSearch(int arr[], int n, int key){  int count=0;  for(int i=0; i<n;i++){  if(arr[i]==key){  count++;  printf("Found at index %d\n", i);  }  }  if(count==0)  printf("KEY NOT FOUND");  printf("TOTAL SEARCH COUNT: %d\n", count);  }  int rec\_linearSearch(int arr[], int n, int key){  if(n-1 < 0)  return -1;  if(arr[n-1]==key)  return n-1;  else return rec\_linearSearch(arr, n-1, key);  }  **B. Implement following searching algorithms in menu:**  **1. Binary search with iteration**  **2. Binary search with recursion**  #include <stdio.h>  void binarySearch(int arr[], int n, int key);  int rec\_binarySearch(int arr[], int n, int key, int low, int high);  int main(){       printf("Enter size of an array: ");      int n, key, count=0;      scanf("%d", &n);      int arr[n];      printf("Enter the elements\n");      for(int i=0; i<n; i++)          scanf("%d", &arr[i]);      printf("Enter the key value to be searched: ");      scanf("%d", &key);        printf("Enter 1: To implement binary search with Iteration\nEnter 2: To implement binary search with Recursion\nEnter your choice:");      int ch;      scanf("%d", &ch);      switch(ch){          case 1: {              printf("USING ITERATION: \n");              binarySearch(arr, n, key);              break;          }          case 2: {              printf("USING RECURSION: \n");              if(rec\_binarySearch(arr, n, key, 0, n-1)==-1)                  printf("KEY NOT FOUND\n");              else                  printf("KEY FOUND AT INDEX: %d", rec\_binarySearch(arr, n, key, 0, n-1));          }      }  }  void binarySearch(int arr[], int n, int key){      int low=0, high=n-1, mid;      int count=0;      while(low<=high){          mid=(low+high)/2;          if(key==arr[mid]){              printf("Key found at index %d\n", mid);              return;          }          else if(key>arr[mid])              low=mid+1;          else              high=mid-1;      }      printf("KEY NOT FOUND\n");  }  int rec\_binarySearch(int arr[], int n, int key, int low, int high){      if(low>high){          return -1;      }      int mid=(low+high)/2;      if(arr[mid]==key)          return mid;      else if(key>arr[mid])          return rec\_binarySearch(arr, n, key, mid+1, high);      else          return rec\_binarySearch(arr, n, key, low, mid-1);  } |
| **Output Screenshots** | **A. Implement following searching algorithm: Linear search with multiple occurrences**    **B. Implement following searching algorithms in menu:**  **1. Binary search with iteration**  **2. Binary search with recursion** |
| **Practice questions** | 1. What is Fibonacci Search explain in detail 2. Write algorithm for Fibonacci search 3. Implement Fibonacci Search 4. o/p screenshot |
| **Conclusion** | Thus we have studied different sorting algorithms and their time complexities. |