**Practical File**

of

**Data Structure and Algorithms Lab**

**(PCC-CS-303)**

submitted in partial fulfillment of the requirement for the award of degree of

**Bachelor of Technology (B.Tech)**

in

**Computer Engineering**

by

**Anuj Sharma**

**(21001003015)**

Under the guidance of

**Mr. Piyush Gupta**

**Assistant Professor**



**Department of Computer Engineering**

**J. C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY, YMCA**

**SECTOR-6 FARIDABAD**

**HARYANA-121006**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Problem** | **List of Programs** | **Date** |
|  | Searching | Given an array of integers *nums* and an integer *target*, write a function to search *target* in *nums*. If *target* exists, then return its index. Otherwise, return *-1*. | 12-10-2022 |
|  | Given an array of integers *nums* which is sorted in ascending order, and an integer *target*, write a function to search target in *nums*. If target exists, then return its index. Otherwise, return *-1*. | 12-10-2022 |
|  | Given a sorted array of *n* elements, possibly with duplicates, find the number of occurrences of the *target* element. | 12-10-2022 |
|  | Given a **0-indexed** integer array *nums*, find a **peak element**, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.  \*A peak element is an element that is strictly greater than its neighbors | 12-10-2022 |
|  | There is an integer array *nums* sorted in ascending order (with distinct values). After the possible **rotation** of the given array, find an integer *target*, return the index of *target* if it is in *nums*, or -1 if it is not in *nums*. | 12-10-2022 |
|  | Given an array *arr* of positive integers sorted in a strictly increasing order, and an integer *k*.  Write a function to return the *kth* positive integer that is missing from this array. | 12-10-2022 |
|  | Stack | Write a program to implement stack using array (Show all the operations like insertion, deletion and display) | 02-11-2022 |
|  | Write a program to convert Infix expression into Postfix expression and also analyze its Complexity. | 02-11-2022 |
|  | Write a program to evaluate the Postfix expression. | 02-11-2022 |
|  | Queue | Write a program to implement Simple Queue using arrays (Show all the operations like insertion, deletion and display) | 16-11-2022 |
|  | Write a program to implement Circular Queue using arrays (Show all the operations like insertion, deletion and display) | 16-11-2022 |
|  | Write a program to implement Priority Queue using both ordered and unordered arrays (Show all the operations like insertion, deletion and display) | 16-11-2022 |
|  | Linked List | Write a Program to insert and delete a node  (i) at the beginning  (ii) at the end  (iii) at any specific position of a singly linked list. | 23-11-2022 |
|  | Write a program to implement linked representation of Stack and Queue. | 23-11-2022 |
|  | Write a Program to insert and delete a node  (i) at the beginning  (ii) at the end  (iii) at any specific position of a Doubly linked list. | 23-11-2022 |
|  | Write a program to search an element in a given singly linked list. | 30-11-2022 |
|  | Write a program to search an element in a given doubly linked list. | 30-11-2022 |
|  | Write a Program to insert and delete a node  (i) at the beginning  (ii) at the end  (iii) at any specific position of a Circular linked list. | 30-11-2022 |
|  | Write a program to search an element in a given Circular linked list. | 07-12-2022 |
|  | Trees | Write a program to implement Binary Search Tree and its operations like insertion, deletion and searching. | 07-12-2022 |
|  | Write a program to traverse Binary Search Tree. | 07-12-2022 |
|  | Write a program to implement AVL tree and its operations like insertion and deletion. | 14-12-2022 |
|  | Sorting | Write a program to implement Selection sort. | 21-12-2022 |
|  | Write a program to implement Bubble sort. | 21-12-2022 |
|  | Write a program to implement Insertion sort. | 21-12-2022 |
|  | Write a program to implement Merge sort. | 28-12-2022 |
|  | Write a program to implement Quick sort. | 28-12-2022 |
|  | Write a program to implement Heap sort. | 28-12-2022 |
|  | Graph | Write a program to implement Breadth First Search traversal technique. | 04-01-2023 |
|  | Write a program to implement Depth First Search traversal technique. | 04-01-2023 |

**Searching**

**Program No. 1:** Given an array of integers *nums* and an integer *target*, write a function to search *target* in *nums*. If *target* exists, then return its index. Otherwise, return *-1*.

**Code:**

#include <bits/stdc++.h>

#define N 9

using **namespace** std;

**int** linear\_search(**int\*** A,**int** key){

    for(**int** i=0;i<N;i++){

        if(A[i]==key) return i;

    }

    return 0;

}

**int** main(){

**int** A[N] = {1,4,3,24,5,26,64,11,9};

**int** key;

    cout << "Enter the value to search for: ";

    cin >> key;

**int** flag = linear\_search(A,key);

    if(flag){

        cout << "element found at index " << flag << endl;

    }else{

        return -1;

    }

}

**Input:**

Enter the value to search for: 11

**Output:**

**element found at index 7**

**Complexity:**

**O(N)**

**Program No. 2:** Given an array of integers *nums* which is sorted in ascending order, and an integer *target*, write a function to search target in *nums*. If target exists, then return its index. Otherwise, return *-1*.

**Code:**

#include <bits/stdc++.h>

#define N 9

using **namespace** std;

**int** binary\_search(**int\*** A,**int** key){

**int** start=0,end=N-1;

    while(start!=end){

**int** mid=(start+end)/2;

        if(A[mid]==key) return mid;

        else if(A[mid]>key){

            end=mid-1;

        }else if(A[mid]<key){

            start=mid+1;

        }

    }

    return -1;

}

**int** main(){

**int** A[N] = {1,2,3,4,5,6,7,8,9};

**int** key;

    cout << "Enter the value to search for: ";

    cin >> key;

**int** flag = binary\_search(A,key);

    if(flag){

        cout << "element found at index " << flag << endl;

    }else if(flag==-1){

        return -1;

    }

}

**Input:**

Enter the value to search for: 3

**Output:**

**element found at index 2**

**Complexity:**

**O(logN)**

**Program No. 3:** Given a sorted array of *n* elements, possibly with duplicates, find the number of occurrences of the *target* element.

**Code:**

#include <bits/stdc++.h>

#define N 9

using **namespace** std;

**int** binary\_search(**int\*** A,**int** key){

**int** start=0,end=N-1;

    while(start!=end){

**int** mid=(start+end)/2;

        if(A[mid]==key)return mid;

        else if(A[mid]>key){

            end=mid-1;

        }else if(A[mid]<key){

            start=mid+1;

        }

    }

    return -1;

}

**int** countOccurrences(**int** **\***A, **int** key)

{

**int** ind = binary\_search(A, key);

    if (ind == -1)

        return 0;

**int** count = 1;

**int** left = ind - 1;

    while (left >= 0 && A[left] == key)

        count++, left--;

**int** right = ind + 1;

    while (right < N && A[right] == key)

        count++, right++;

    return count;

}

**int** main(){

**int** A[N] = {1,2,3,3,3,3,7,8,9};

**int** key;

    cout << "Enter the value to search for: ";

    cin >> key;

**int** flag = countOccurrences(A,key);

    cout << "Number of occurences: " << flag << endl;

}

**Input:**

Enter the value to search for: 3

**Output:**

**Number of occurences: 4**

**Complexity:**

**O(logN)**

**Program No. 4:** Given a **0-indexed** integer array *nums*, find a **peak element**, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

\*A peak element is an element that is strictly greater than its neighbors.

**Code:**

**Input:**

**Output:**

**Complexity:**

**Program No. 5:** There is an integer array *nums* sorted in ascending order (with distinct values). After the possible **rotation** of the given array, find an integer *target*, return the index of *target* if it is in *nums*, or -1 if it is not in *nums*.

**Code:**

**Input:**

**Output:**

**Complexity:**

**Program No. 6:** Given an array *arr* of positive integers sorted in a strictly increasing order, and an integer *k*. Write a function to return the *kth* positive integer that is missing from this array.

**Code:**

**Input:**

**Output:**

**Complexity:**

**Program No. 7:** Write a program to implement stack using array (Show all the operations like insertion, deletion and display)

**Code:**

**Input:**

**Output:**

**Complexity:**

**Program No. 8:** Write a program to convert Infix expression into Postfix expression and also analyze its Complexity.

**Code:**

**Input:**

**Output:**

**Complexity:**

**Program No. 9:** Write a program to evaluate the Postfix expression.

**Code:**

**Input:**

**Output:**

**Complexity:**

**Program No. 10:** Write a program to implement Simple Queue using arrays (Show all the operations like insertion, deletion and display)

**Code:**

**Input:**

**Output:**

**Complexity:**

**Program No. 11:** Write a program to implement Circular Queue using arrays (Show all the operations like insertion, deletion and display)

**Code:**

**Input:**

**Output:**

**Complexity:**

**Program No. 12:** Write a program to implement Priority Queue using both ordered and unordered arrays (Show all the operations like insertion, deletion and display)

**Code:**

**Input:**

**Output:**

**Complexity:**