Logo, company name

Description automatically generated

Name: Manikanta Mandala.

Sec: CSE-3.

Enrolment No. : 200C2030134

Problem Statement:

Given a sorted array A[] with possibly duplicate elements, write a program to find indexes of first and last occurrences of an element k in the given array.  
Problem Note

* Perform the time complexity of linear and binary search
* How to attain the algorithm’s runtime complexity in the order of O(log n) (Do analysis and suggest possible solution)
* If the target is not found in the array, return [-1, -1].

Example : A[ ] = [1, 3, 5, 5, 5, 5 ,28, 37, 42], target = 5   
 Output: [2, 5]

Solution Approach(Binary Search):

1. Write the Binary Search function for searching for both first and last.
2. Modify the scope of if condition { if(a[mid]==key) } as
3. Make a condition if(arr[mid-1]!=key) then return mid value for binaryFirstOccurance and {if(arr[mid+1]!=key)} then return mid value for binaryLastOccurance
4. If the condition does not satisfy then shift last value to left in binaryFirstOccurance and shift first value to right in binaryLastOccurance respectively.

3. write main function to take size, array elements and the key and passing the values to respective functions

4. Printing the first and last occurrence.

Solution Approach(Linear Search):

1. Write the Linear Search function for searching for both first and last.
2. There is no big change in linear search. Just applying loop with the condition i<n.
3. In this loop, there should be another condition like if(arr[mid]==key) then return i else return -1

Code:

#include<stdio.h>

int linearFirstOccurance(int arr[],int n, int key){

for(int i=0;i<n;i++){

if(arr[i]==key){

return i;

}

}

return -1;

}

int linearLastOccurance(int arr[],int n, int key){

for(int i=n-1;i>=0;i--){

if(arr[i]==key){

return i;

}

}

return -1;

}

int binaryFirstOccurance(int arr[],int n,int key){

int f=0,l=n-1,mid;

while(f<=l){

mid=(f+l)/2;

if(arr[mid]==key){

if(arr[mid-1]!=key){

return mid;

}

l=mid-1;

}

else if(arr[mid]>key){

l=mid-1;

}

else{

f=mid+1;

}

}

return -1;

}

int binaryLastOccurance(int arr[],int n,int key){

int f=0,l=n-1,mid;

while(f<=l){

mid=(f+l)/2;

if(arr[mid]==key){

if(arr[mid+1]!=key){

return mid;

}

f=mid+1;

}

else if(arr[mid]>key){

l=mid-1;

}

else{

f=mid+1;

}

}

return -1;

}

int main(){

int n,first,last,key;

printf("Enter the size ");

scanf("%d",&n);

int arr[n];

printf("Enter the elements in array ");

for(int i=0;i<n;i++){

scanf("%d",&arr[i]);

}

printf("Enter the key ");

scanf("%d",&key);

/\*

first=linearFirstOccurance(arr,n,key);

last=linearLastOccurance(arr,n,key);

\*/

first=binaryFirstOccurance(arr,n,key);

last=binaryLastOccurance(arr,n,key);

printf("[%d,%d]",first,last);

return 0;

}

Input and output for Linear searching:

A screenshot of a computer

Description automatically generated with medium confidence

Input and output for binary searching:

A picture containing text, monitor, screenshot, screen

Description automatically generated

Time Complexity:

🡺Time complexity of binary search will be O(log2(n)) 🡺log n base 2.

🡺Time complexity of linear search will be O(n).

To attain the algorithm’s runtime complexity in order of O(log n) we need to use binary search.