Name : Atharva Dhage

Roll No : 231070017

Branch: Computer Engineering

DAA Laboratory

4 Algorithm

• Linear search Algorithm

Algorithm
Binney Search
11
// Limear Saarch
Brown Seas (water) deed grands
Linear Search (vector < int > arr, Key)
eres est as yest a brief at the const
Search an alement / Kyz: search a key ien the arro
Input: Ky and over is taken as comput.
Output: 95 key as found in arr the given the
index otherwise -1
Toront = and o= travet
i=0 ans;
while i < arm size do
gg arr [i] = Key
veturn as ans
11 39 dorap and without king finding key
outurn falsa - 1;
elseif our Enid] = Key
1 th ford = 1 travel

• Binary Search

	D's a l
	Birnary Search
	William Seascon
	Bineary Search (vector < int > arr, key)
	Circus Sinth (retorsint) and, Key)
	Search on ky: to find a key in the over
15	Input: les doke arr, key as a comput
	Output: If ky in present then octure induce
- 1947	alluwise ge creture -1: 100
	(saturates their
	Front =0, back = arr. size -1
	; and 0=1
	where while front < back do
	mid = (grant + back)/2
	and the multiple
	are [mid] = they
	: Locature mid:
	elseif arr [mid] < Key
	front f = mid+1
	else
	back b= mid-1
	// if + cloop and without key
	Juturn -1

← Code

• Linear Search

```
#include <bits/stdc++.h>
using namespace std;
// Linear Search
int LinearSearch(vector<int> arr, int key)
    int i = 0;
    while (i < arr.size())</pre>
        if (arr[i] == key)
             return i;
        i++;
int main()
    int key, size, a;
    cout << "Enter size";</pre>
    cin >> size;
    vector<int> v(size);
    cout << "Enter the key";</pre>
    cin >> key;
    cout << "Enter the element";</pre>
    for (int i = 0; i < size; i++)</pre>
        cin >> a;
        v[i] = a;
    cout << endl;</pre>
    cout << "Output--> :" << LinearSearch(v, key);</pre>
    return 0;
```

• Binary Search

```
#include <bits/stdc++.h>
using namespace std;
// Binary Search
int BinarySearch(vector<int> arr, int key)
    int b = arr.size() - 1;
    int mid;
    while (f <= b)
    {
        mid = (f + b) / 2;
        if (arr[mid] == key)
        {
             return mid;
        else if (arr[mid] < key)</pre>
            f = mid + 1;
        {
            b = mid - 1;
int main()
    int key, size, a;
    cout << "Enter size";</pre>
    cin >> size;
    vector<int> v(size);
    cout << "Enter the key";</pre>
    cin >> key;
    cout << "Enter the element";</pre>
    for (int i = 0; i < size; i++)</pre>
        cin >> a;
        v[i] = a;
    cout << "Output--> " << BinarySearch(v, key);</pre>
    return 0;
```

Testcases

	Text		
	Test - case		
	Linear Search		autput
	Input		anguer
	aveay	Kuy	Experted + output
)	\$ 1,2,3,4,53	4	3
2)	Ş-1,-5, 6,73	-5	1
3)	2-1, 0, 333, 14,10}	2	-1
4)	§ -11 , -12, 13, -14, 15}	30	-1
5)	§ 3	55	-1
	Rineary Search		
)	2 1, 2, 3, 4, 53	ч	28 3
2)	\$ 60, 61, 623	62	2
3)	§ 1003	100	0
4)	5-4,-3,-2,-13	5	-1
5)	§ 10, 113	933	

• My-Output

• Linear Search

```
1)
Enter size 5
Enter the element1 2 3 4 5
Enter the key 4
Output--> 3
2)
Enter size 4
Enter the element -1 -5 6 7
Enter the key -5
Output--> 1
3)
Enter size5
Enter the element -1 0 333 14 10
Enter the key 2
Output--> -1
Enter size : 5
Enter the element : -11 -12 13 -14 15
Enter the key : 30
Output--> -1
5)
Enter size : 0
Enter the element : Enter the key : 55
Output--> -1
```

• Binary Search

```
1)
Enter size : 5
Enter the element :1 2 3 4 5
Enter the key : 4
Output--> 3
2)
Enter size : 3
Enter the element :60 61 62
Enter the key : 62
Output--> 2
3)
Enter size : 1
Enter the element :100
Enter the key : 100
Output--> 0
Enter size : 4
Enter the element :-4 -3 -2 -1
Enter the key : 5
Output--> -1
5)
Enter size : 2
Enter the element :10 11
Enter the key : 999
Output--> -1
```

• Time Complexity

Persona Design
Time composition to make the confusion (12)
tinear search
Best care :- D(n) = 1/9 Key in at 0 th index Worst case :- O(n) 1/ as there are n element g Key is in dost position
average cose: - O(n) etisticted monatal (1) (aparoness position) man 3 = 12
Binary (sewish dietz mittad) mand
But cuse: - O(A) dogsens O(dogs) Growst cuse :- O(A) dogsens O(dogs)
So = & more (no of extentialin)

• Conclusion

Linear search checks each element sequentially and is best for small or unsorted datasets, with a time complexity of O(n).

Binary search, on the other hand, is more efficient for large, sorted datasets, with a time complexity of O(log n).

.The choice between them depends on the dataset's size and whether it is sorted.