Name : Atharva Dhage

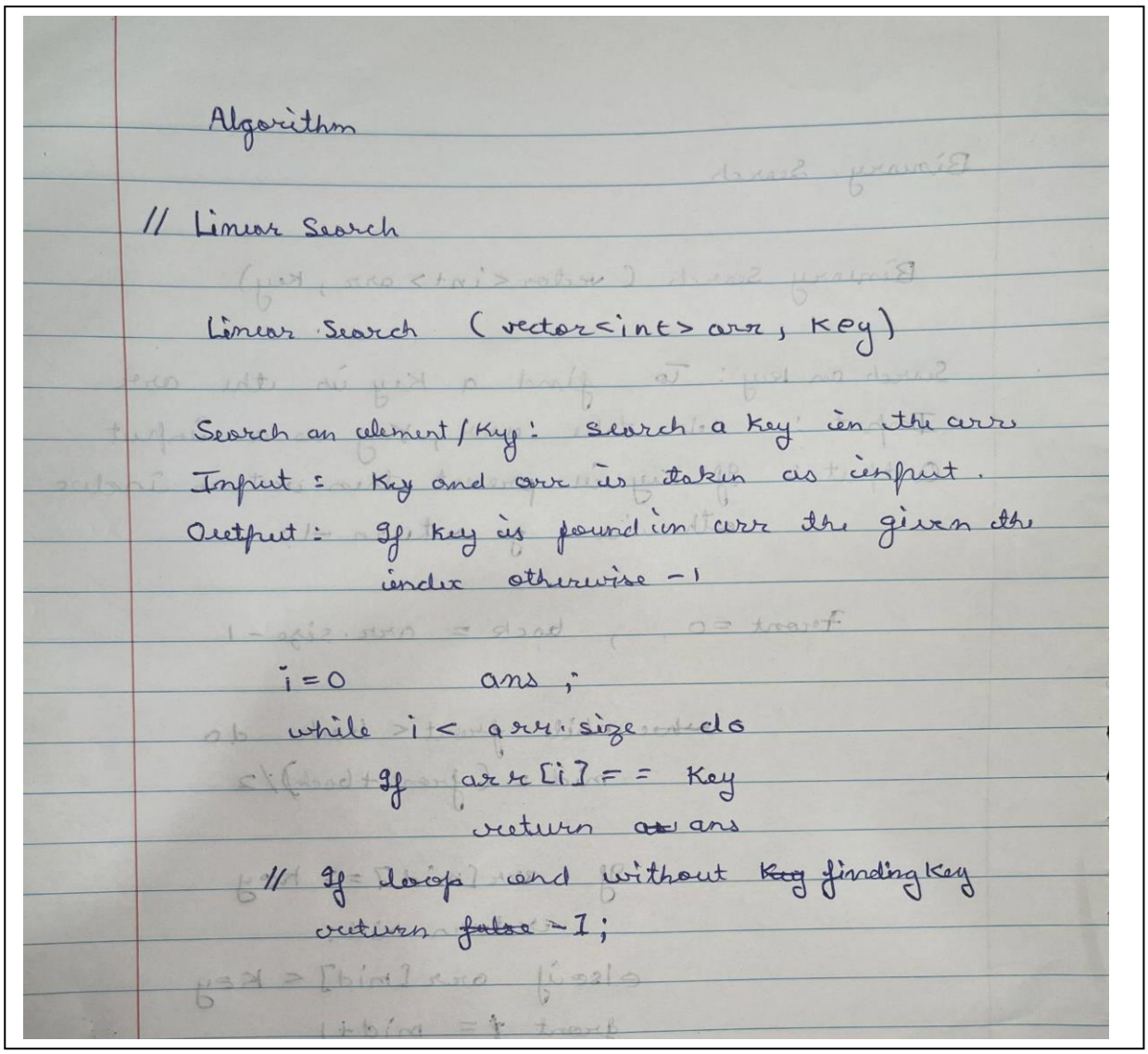
Roll No : 231070017

Branch : Computer Engineering

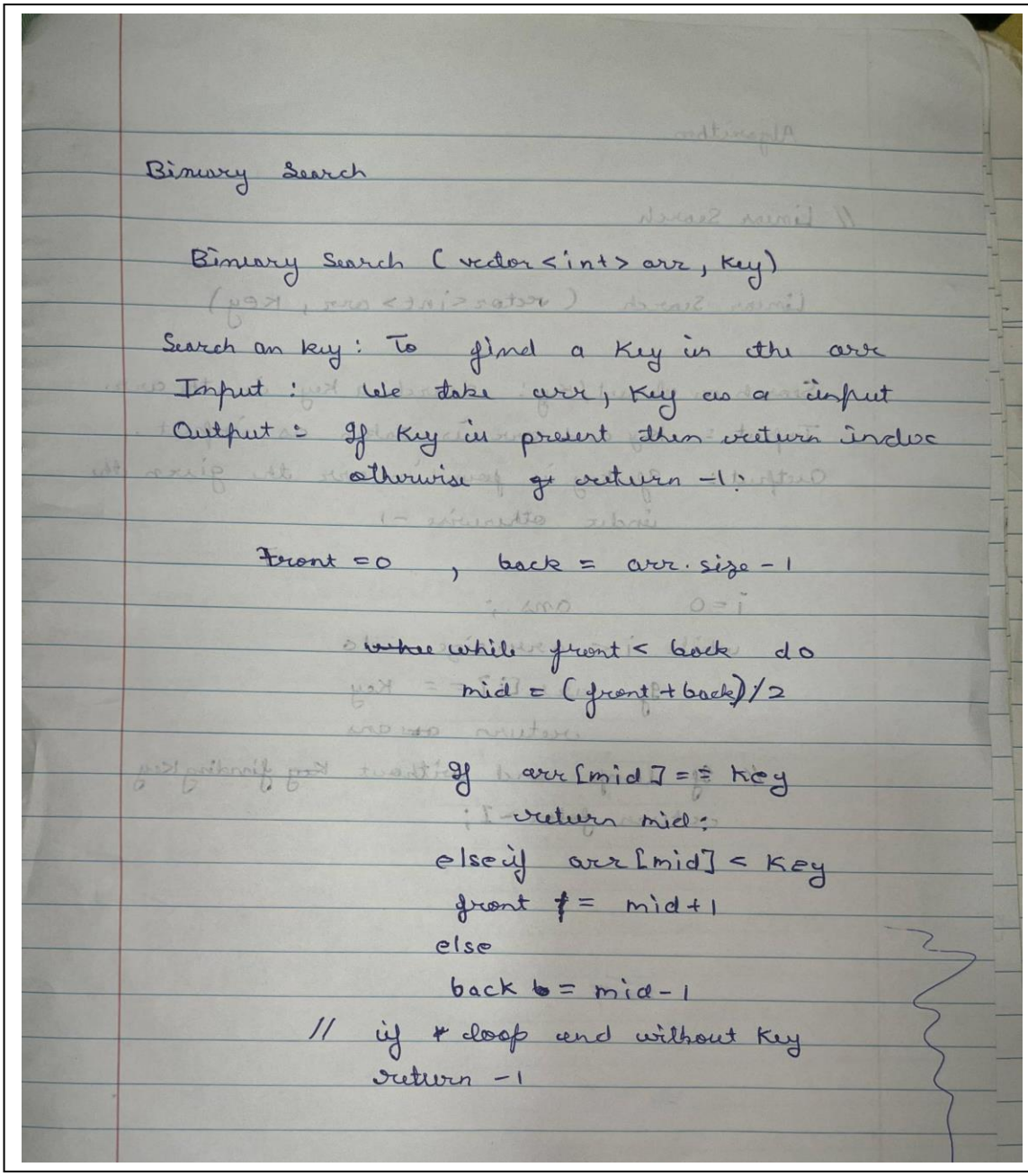
DAA Laboratory

Algorithm

- Linear search Algorithm



- Binary Search



Binary Search

Binary Search (vector<int> arr , Key)

Search an key : To find a Key in the arr
Input : We take arr, Key as a input
Output : If Key is present then return index
otherwise it return -1 :

Front = 0 , back = arr.size - 1

while front < back do
mid = (front + back)/2

If arr[mid] == Key
return mid ;
elseif arr[mid] < Key
front = mid + 1
else
back = mid - 1
// if * loop end without Key
return -1

Code

- Linear Search

```cpp
#include <bits/stdc++.h>
using namespace std;

// Linear Search
int LinearSearch(vector<int> arr, int key)
{
    int i = 0;
    while (i < arr.size())
    {
        if (arr[i] == key)
        {
            return i;
        }
        i++;
    }
    return -1;
}

int main()
{
    int key, size, a;
    cout << "Enter size";
    cin >> size;
    vector<int> v(size);
    cout << "Enter the key";
    cin >> key;
    cout << "Enter the element";
    for (int i = 0; i < size; i++)
    {
        cin >> a;
        v[i] = a;
    }
    cout << endl;
    cout << "Output--> :" << LinearSearch(v, key);

    return 0;
}
```

- **Binary Search**

```cpp
#include <bits/stdc++.h>
using namespace std;

// Binary Search

int BinarySearch(vector<int> arr, int key)
{
    int f = 0;
    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)
        {
            f = mid + 1;
        }
        else
        {
            b = mid - 1;
        }
    }
    return -1;
}

int main()
{
    int key, size, a;
    cout << "Enter size";
    cin >> size;
    vector<int> v(size);
    cout << "Enter the key";
    cin >> key;
    cout << "Enter the element";
    for (int i = 0; i < size; i++)
    {
        cin >> a;
        v[i] = a;
    }
    cout << "Output--> " << BinarySearch(v, key);

    return 0;
```

Testcases

Test - case

Linear Search

	Input: array	Key	Output: Expected output
1)	{1,2,3,4,5}	4	3
2)	{-1,-5, 6, 7}	-5	1
3)	{-1, 0, 333, 14, 10}	2	-1
4)	{-11, -12, 13, -14, 15}	30	-1
5)	{}	55	-1

Binary Search

1)	{1, 2, 3, 4, 5}	4	3
2)	{60, 61, 62}	62	2
3)	{100}	100	0
4)	{-4, -3, -2, -1}	5	-1
5)	{10, 11}	999	-1

- My-Output

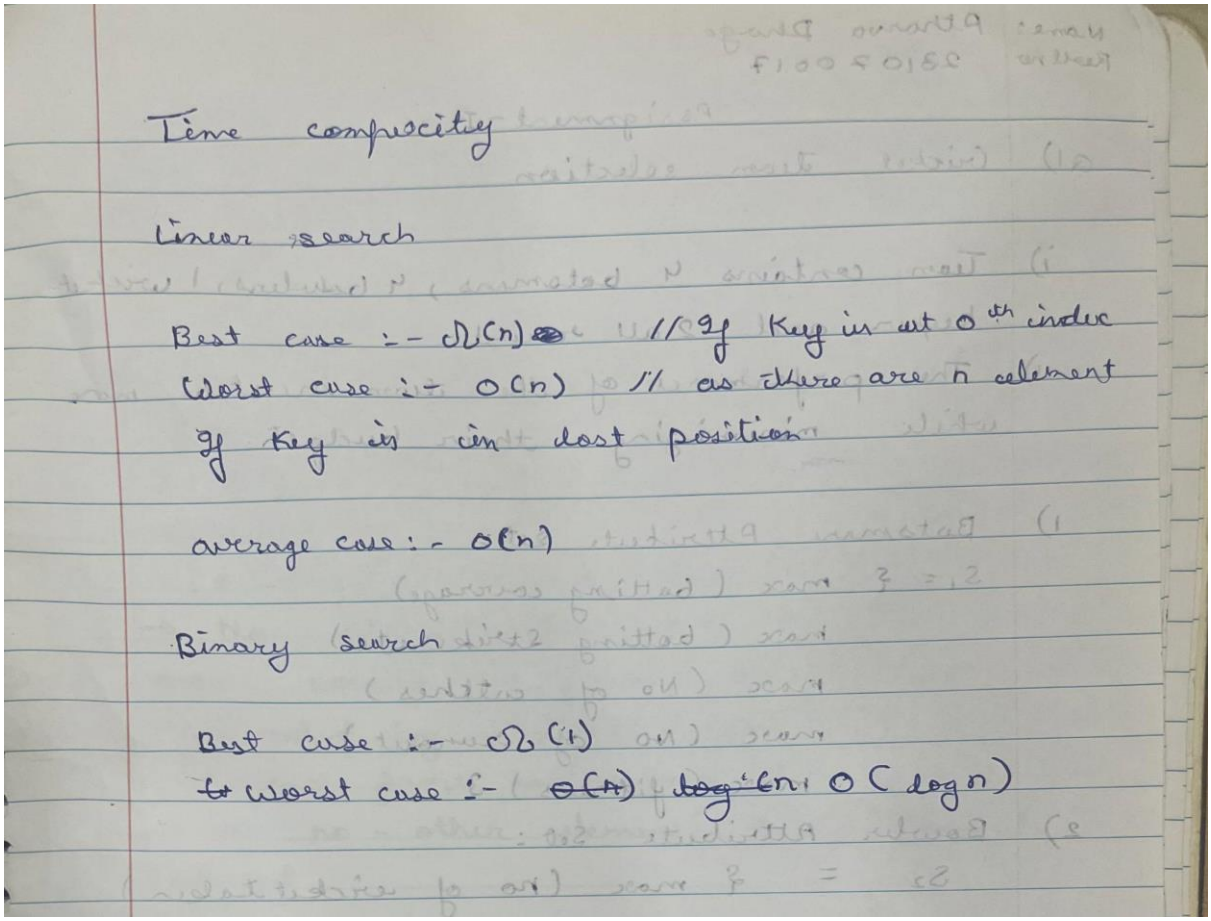
- Linear Search

```
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
4)
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
4)
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



Time complexity

Linear search

Best case :- $\Omega(n)$ // If Key is at 0th index

Worst case :- $O(n)$ // as there are n element if Key is in last position

average case :- $O(n)$

Binary search

Best case :- $\Omega(1)$

Worst case :- ~~$O(n)$~~ ~~$\log(n)$~~ $O(\log n)$

- 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.