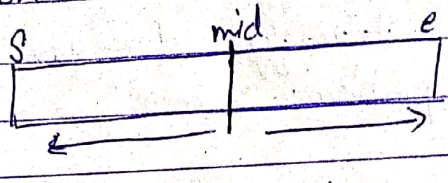
Day-2 Hello ☺ R, U. ready ?
Yes
Let's Go !

Binary Search

Find a index of element in sorted array using Recursion.



In this type of question, we divide an array into subparts.
It is called as "Divide and Conquer".

Types of Recursion :-

(1) Linearly :- Example, fibonaccie. It takes a lot time & memory.
(2) Divide & Concure :- Example, Binary search. It is a benificial.

✓ Step 1 :- Find recurion is applicable or not.

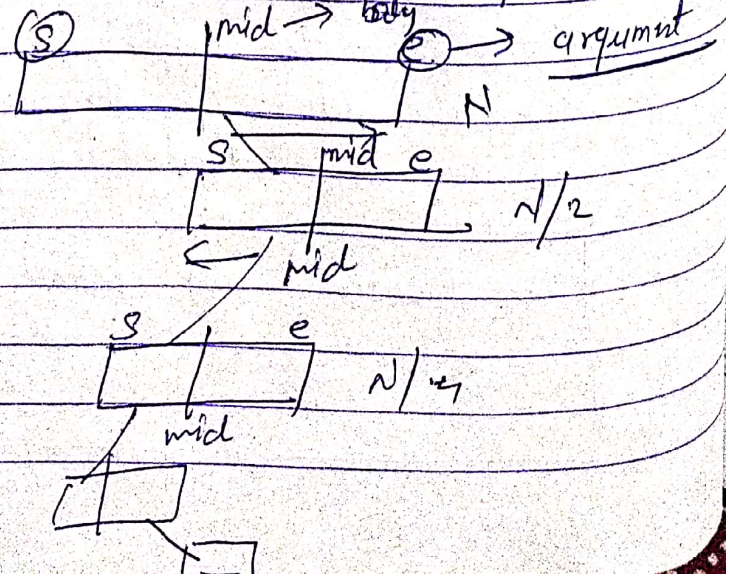
✓ Step 2 :- Find Recurrance Relation :-

mid divide

$$Search(n) = O(1) + Search(N/2)$$

Step 3 :- Tree

✓



Important Tip :-

Step 5 About the variables :-

Three places where we have to put right type of variables →

1) Argument 2) Body 3) Return

(1) Argument variable are those, who are going to help us in future calls.

like binary search, we can't divide array without "start and end" variable.

So, "start and end" variable will be put inside the function argument.

(2) Body Part :- Mid value is only applicable for that particular function.

That is why, we put this "mid" value inside the body of function

(3) Return Part :- For the return part, we are returning mid that

Return is simple as normal function return part.

Code:-

```
int main ()
{
    vector arr = [1|2|3|4|5|6]
    int target = 4
    int start = 0, end = arr.size() - 1;
    search ( arr, target, start, end);
}
```

Recursion function

```
int search ( arr, target, start, end)
{
    if ( s > e)                    // Base condition
    {
        return -1;
    }
    if ( arr[mid] == target)       // getting ans
        return mid;

dividing part
    if ( arr[mid] < target)
        search ( arr, target, mid+1, end);

    if ( arr[mid] > target)
        search ( arr, target, start, mid-1);
}
```

Now, It's time to Practice! ☺

② Sum triangle from array :-

I/p :- A = {1, 2, 3, 4, 5}

O/p :-
[4, 8] l = 5
[20, 28] l = 4
[8, 12, 16] l = 3
[3, 5, 7, 9] l = 2
[1, 2, 3, 4, 5] l = 1

sol →

0	1	2	3	4
1	2	3	4	5

Step 1 :- Recursion is applicable, bcoz, initially we have the whole array and after that we are shorting array with some condition. So, it's creating same thing like binary search.

Step 2 →

1	2	3	4	5
0	1			

f(0) + f(1) + f(1) + f(2)

3	5

1) Argument
2) Body
3) Return

S : 1, 2, 3, 4, 5

arrsum(arr, s(0), s

0	1	2	3	4
1	2	3	4	5

body → f(0) + f(1) f(1) + f(2) f(2) + f(3) f(3) + f(4)

3	5	7	9

↳ next step call

base (n < 1)
return.

Code :-

```
int main()
{
    vector<int> arr = {1, 2, 3, 4, 5};
    int n = arr.size();

    printTriangle(arr, n-1);
}

void printTriangle(vector<int>& arr, int n)
{
    if(n < 1)          // Base condition
    {
        return;
    }
// body
    vector<int> temp(n-1);

    for(int i=0; i<n; i++)
    {
        int x = arr[i] + arr[i+1];
        temp.push_back(temp.begin()+i, x);
    }

    printTriangle(temp, n-1);

    for(int i=0; i<n; i++)
    {
        if(i == n-1)
        {
            cout << arr[i] << " ";
        }
        else
        {
            cout << arr[i] << ", ";
        }
    }
    cout << endl;
}
```

Program Work flow in Stack