Day-14

Binary Search:

Used for sorted arrays

arr = [2, 4, 9, 10, 12, 14, 18] → Asc

arr2 = [18, 14, 12, 6, -7, -18] → Des

Target = 14

1. Find the middle element
2. Check
   If Target > middle => Search in right else in left.
3. If target == middle => We found element

Eg: arr = [2, 4, 6, 9, 11, 12, 14, 20, 36, 48] Target = 36

1. Find middle element → Index value

$$mid = \frac{start + end}{2} = \frac{0+9}{2} = 4$$

2nd check · Is target > middle = 36 > 11 Yes → Then check in right half

Now arr = [2, 4, 6, 9, 11, 12, 14, 20, 36, 48]

From 5th Index search mid value $\frac{5+9}{2} = 7$

Again Check:

Target > middle => 36 > 20 then right half

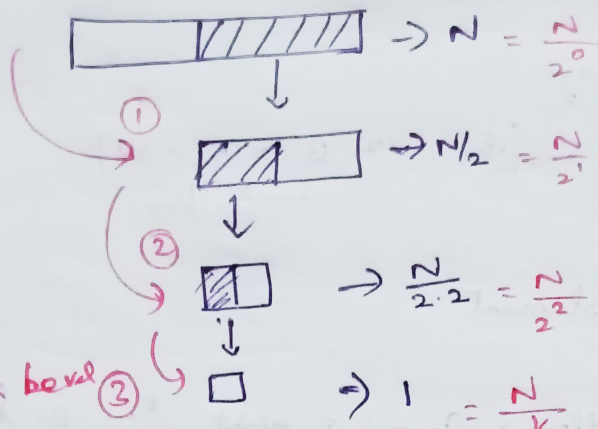
(36), 48 Mid = $\frac{8+9}{2} = 8$ [Check 8th Index]

Target == middle => 36 == 36 We found the element at Index 8

If target > middle → Right check → Take middle value & repeat

Target < middle → Left check → Take middle value & repeat the process

Why Binary Search?



$$\frac{N}{2^k} = 1$$

$$N = 2^k$$

$$\log(N) = \log(2^k)$$

$$\log N = k \log 2$$

$$k = \frac{\log N}{\log 2}$$

$$\boxed{k = \log_2 N}$$ → Size of array

k → Total no of comparison in worst case

Total comparison in worst case = $\log N$

best case = $O(\log N)$

Better way to find mid:

→ mid = $\frac{start + end}{2}$ If large value may exceed int range

→ mid = $start + \frac{(end - start)}{2}$

Explanation:

$$s + \frac{(e - s)}{2}$$

$$\frac{2s + e - s}{2} = \frac{s + e}{2}$$

Program:

```
public class binary{
    public static void main (String [] args){
        int [] arr = {-1, 0, 3, 6, 7, 9};
        int target = 7;
        System.out.println ( binarysearch (arr, target));
    }
    static int binarysearch ( int [] arr, int target){
        int start = 0;
        int end = arr.length - 1;
        while ( start <= end){
            int mid = start + (end - start) /2;
            if ( target < arr[mid] ){
```

```
            end = mid - 1;
        }
        else if ( target > arr[mid]){
            start = mid + 1;
        }
        else {
            return mid ;
        }
    }
    return -1;
  }
}
```

o/p: 4

NOTE: Given arr should be sorted array

If target = 8
o/p -1

Order Agnostic Binary Search

```
public class orderagnbs {
    public static void main (String[]args){
        int [] arr = {-1,0,3,6,7,9};
                      0 1 2 3 4 5
        int target = 7;
        S.o.p (orderAgnbs(arr, target));
    }
    static int orderAgnbs (int [] arr,
                           int target){
        int start = 0 ;
        int end = arr.length -1 ;
        boolean isAsc = arr[start] <
                        arr[end];
        while ( start <= end){
            int mid = start + (end -
                      start) / 2;
            if (arr[mid] == target){
                return mid ;
            }
            if ( isAsc ){
                if (target < arr[mid]){
                    end = mid - 1 ;
                }
                else {
                    start = mid + 1 ;
                }
            }
            else {
                if (target > arr[mid]){
                    end = mid - 1;
                }
                else {
                    start = mid + 1;
                }
            }
        }
        return -1;
    }
}
```

Start < end → Asc
Start > end → Des

o/p 4

If int [] arr = {99, 7, 6, 5, 4, 2}
0 1 2 3 4 5 6

o/p: 2