• Binary Search :-

Q→ Why binary search ?

ans→ The binary search algorithm uses the divide and conquer approach, it does not scan every element in the array, it only search half of the list instead of going through each element. hence it is said to be best searching algorithm.

Ex→ array of 10,00,000 elements.
Linear search will take 1000000 comparison in worst case. But binary search will take only 20 comparison. because $\log_2 1000000 = 20$

worst case ⇒ $O(\log N)$.

• Logic and algorithm :- (array in ascending order)

i> Find the middle index.
ii> Compare it with target value.
iii> if target > middle → Search in the right side
iv> else in the left side.
v> if target == middle → Sout middle

middle
arr = [2, 4, 8, 10, (12,) 14, 16, 18, 20]

target → 14 →

$middle = s + \frac{(end - s)}{2}$

since target > middle

So search in the right side.

arr = [2, 4, 6, 8, 10, (12), 14, 16, (18) 20]
start, end, new middle

now target < middle
So our end will be shifted. (end = m-1)

arr = [2, 4, 6, 8, 10, 12, (14), 16, 18, 20]
start, end, middle

Since target == middle
↳ Return middle

- Better way to find middle element :-

$$\text{Middle} = \text{Start} + \frac{(\text{end} - \text{Start})}{2}$$

because int have some range which might be exceed and we'll be getting errors. That's why we are using like this.

* The one prequisite of binary search is that an array should be in sorted order. It can be in ascending or descending order.

* Code for binary search :- (Order agnostic)

```
public class binarySearch
{
    public static void main (String[] args)
    {
        int[] arr = {2, 4, 6, 8, 10, 12, 16, 27, 31, 1004};
        int target = 27;
        int ans = binarySearch (arr, target);
        Sout (ans);
    }
    Static int binarySearch (int[] arr, int target)
    {
        int start = 0;
        int end = arr.length - 1;
        while (start <= end)
        {
            int middle = start + (end - start) / 2;
            if (arr[middle == = target)
            {
                return middle;
            }
            else if (arr[start] < arr[end])
            {
                if (target < arr[middle])
                {
                    end = middle - 1;
                }
                else if (target
                {
                    start = middle + 1;
                }
```

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

* Practice ques is in tutorial 7 folder.

* Order agnostic is the process to check whether the array is in descending order or in ascending order.