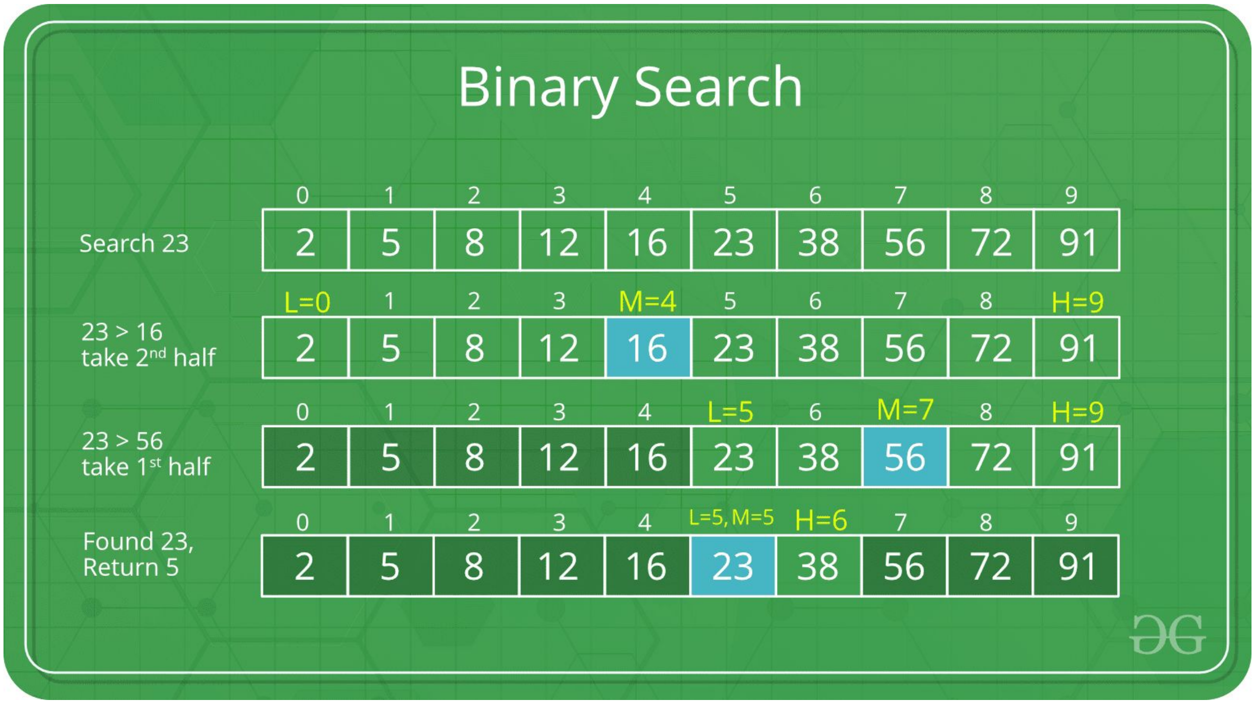
Binary Search :



Complexity : Time & Space

Calculating Time complexity:

- Let say the iteration in Binary Search terminates after **k** iterations. In the above example, it terminates after 3 iterations, so **here k = 3**
- At each iteration, the array is divided by half. So let's say the length of array at any iteration is **n**

- At **Iteration 1**,

  `Length of array = n`
- At **Iteration 2**,

  `Length of array = n⁄2`
- At **Iteration 3**,

  `Length of array = (n⁄2)⁄2 = n⁄22`
- Therefore, after **Iteration k**,

  `Length of array = n⁄2k`
- Also, we know that after

  `After k divisions, the length of array becomes 1`

Therefore

`Length of array = n⁄2k = 1`

`=> n = 2k`

Applying log function on both sides:

`=> log2 (n) = log2 (2k)`

`=> log2 (n) = k log2 (2)`

As **($\log_a (a) = 1$)**

Therefore,

`=> k = log2 (n)`

Hence, the time complexity of Binary Search is

$\log_2 (n)$

Nested-loop complexity :

```
for(i=0; i<n; i++)
{
        for(j=0; j<n; j++)
        {
                for(k=0; k<n; k++)
                {
                }
        }
}
```

i=0, j=0,1,2...n-1
j=0; k=0,......n-1
….
i=1, j=0,1,2...n-1
….
i=n-1, j=0,1,....n-1

n=5, 5*5*5=125

Complexity : big_o(n^2)

Big_o -> Growth Function

2x^3 + x^2 + 3x

Complexity = max(power) = big_o(x^3)

Suppose , nested loop of i,j,k; complexity = 2 * big_o(n^3)
Nested loop i,j ; complexity = big_o(n^2)
loop of i ; complexity = 3 * big_o(n)