IV Analysis of Binary Search

(a) Binary Search

- It devides a large array into two smaller subarrays and then recursively operate the subarray.
- But instead of operating on both subarrays, it discards one sub-array and continue on the second sub-array.
- This decision of discarding is done with just one comparison.
- It works with already sorted array.
- In this, at each step mid value is calculated & compared with target value.
- there are three cases possible:
  1. If target = A[mid], return mid.
  2. If target < A[mid], then search in first half.
  3. If target > A[mid], then search in second half.

(b) Algorithm:

i/p : ~~low~~, start, end → indices of global array a[],
x → element to be searched.

o/p : return index.

```
BINARY_SEARCH ( start, end, x)
{
1.    if (start > end)          // Base condition.
2.        return -1

3.    mid = (start + end)/2     // calculating mid index

4.    if (x == A[mid])          // case 1
5.        return mid.

6.    elseif (x < A[mid])       // case 2.
7.        return BINARY_SEARCH ( start, mid-1, x)

8.    else.                     // case 3.
7.        return BINARYSEARCH ( mid+1, end, x)
```

Initial call to BINARYSEARCH will be BINARYSEARCH (1, a.length, x)

② Analyses of Binary Search

Divede : Algorithm computes middle index i.e. takes constant time ie $O(1)$

Conquer : In this, one subproblem of size $n/2$ is discarded, Hence, we recursively solve one subproblem of size $\lfloor n/2 \rfloor$, which contributes to $T(n/2)$ in running time.

Combine: In this algorithm, returns an index, Hence takes $\Theta(1)$ time

∴ Recurrence for the running time $T(n)$ of Binary search is,

$$T(n) = \begin{cases} \Theta(1) & \text{if } n=1 \\ T\left(\frac{n}{2}\right) + \Theta(1) & \text{if } n>1 \end{cases}$$

Solving this recurrence using Master method

Step 1: Comparing with standard form

i.e $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a = 1 \quad b = 2 \quad f(n) = 1$

$$\log_b a = \log_2 1 = 0$$

$$n^{\log_b a} = n^0 = 1$$

Step 2: Identifying case,

$$f(n) = 1$$
$$= n^{\log_b a}$$
$$= \Theta(n^{\log_b a})$$

By case 2,

$$T(n) = \Theta(n^{\log_b a} \log n)$$
$$= \Theta(n^0 \log n)$$
$$\boxed{T(n) = \Theta(\log n)}$$