# BINARY SEARCH

Date ______
Page ______

→ An array having n elements in increasing order. The problem is to determine whether a given element $x$ is present in the list or not.

→ If the element $x$ is present in the list then we are to determine the index of the list $k$, where $a_k = x$.

→ If $x$ is not in the list then $k$ is set to be zero.

→ Let $P = (n, a_i, \ldots, a_j, x)$ is an instance of search problem.
$n$ is the number of element of elements in the list.
$a_i \cdots a_j$ is the list of elements.
$x$ is the element to be searched.

→ Binary search can be solved with the help of divide-and-conquer.

→ If the problem $P$ has more than one element, it can be divided into new subproblems.

→ Pick up an index $q$ and compare $x$ with $a_q$. There are 3 cases.
(i) If $x = a_q$, then the problem $P$ is immediately solved.
(ii) If $x < a_q$, then $x$ has to be searched for only in the sublist $a_i, a_{i+1}, \ldots, a_{q-1}$
(iii) If $x > a_q$, then $x$ has to be searched for only in the sublist $a_{q+1}, \ldots, a_j$

→ If the problem having only one element, then it takes $\Theta(1)$ time.

→ If the problem having more than one elements then we go for divide and conquer approach.
Then middle element $q = ((n+1)/2)$

→ Here, solution of the new subproblem is the solution of this original problem.

→ There is no need of combining the solutions of the subproblems to get the solution of the original problem.

Algorithm BinarySearch $(a, i, l, x)$
where $a[\,]$ is the array
$i$ is the starting index of the array.
$l$ is the ending index of the array.
$x$ is the item which is to be searched.

```
1.  If (i = l) Then
2.      If (x = a[i]) Then
3.          return i;
4.      else
5.          return 0
```

79

~~BINARYSEARCH(A, i, j, x)~~

```
else
    mid = [(i+l)/2]
    if (x = a[mid]) Then
        return mid
    else if (x < a[mid]) Then
        return BinarySearch(a, i, mid-1, x)
    else
        return BinarySearch(a, mid+1, l, x)
```

**Example:**

| 11 | 22 | 33 | 44 | 55 | 66 | 77 |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| ↑ i | | | ↑ mid | | | ↑ j |

Searching element $x = 33$

~~Is i ≤ j Yes~~ $mid = \frac{0+6}{2} = 3$

Is $33 == a[3]$ NO

$33 < a[3]$

$i = 0$ and $j = mid - 1 = 2$

$mid = \frac{0+2}{2} = 1$

Is $33 == a[1]$ NO

$33 > a[1]$

$i = mid + 1 = 2$, $j = 2$

$mid = \frac{2+2}{2} = 2$

Is $33 = a[2]$ Yes

The search is successful at index 2.

**Analysis Of Binary Search:**

→ We are dividing the problem into two halves and solving only one subproblem not two to get the solution of the original problem. So, the value of a is 1.

→ The number of input size for subproblems at each division is reduced by half. So, the value of b is 2.

→ f(n) is a function which denote the time taken by the problem to divide as well as to combine the solution of the subproblems to get the solution of the original problem.

→ In binary search, we are not spending time for combining the solutions of subproblems.

→ For dividing the original problem we take $\Theta(1)$ time.

→ Now the recurrence can be of form

$$T(n) = \begin{cases} T(1) & n=1 \\ T(n/2) + \Theta(1) & n>1 \end{cases}$$

$$T(n) = T(n/2) + 1$$

$a=1$, $b=2$, $f(n)=1$

$$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$$

Case-2 of master ~~Theorem~~ method is satisfied. So, solⁿ is

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