"रास्ते खुद बन जाते हैं,
हौसलों के कदम जब उठते हैं।"

Binary Search

What is Binary Search

- Works on sorted array
- ✓ DnC
- Search Algo.

Searching algo. which search the data in

Sorted Search Space

$mid = \frac{(s+e)}{2}$

N = 5

x = 12

s

e

s

s = 0

e = 4

m = 2

s = 3
e = 4
m = 3

s = 4
e = 4
m = 4

s = 5
e = 4

a(m)	x
a = b	→ return mid
a < b	→ Right
a > b	→ left

- define search space
- find mid
- check (mid element == target) $a = b$
- if (a(m) < target) $a < b$

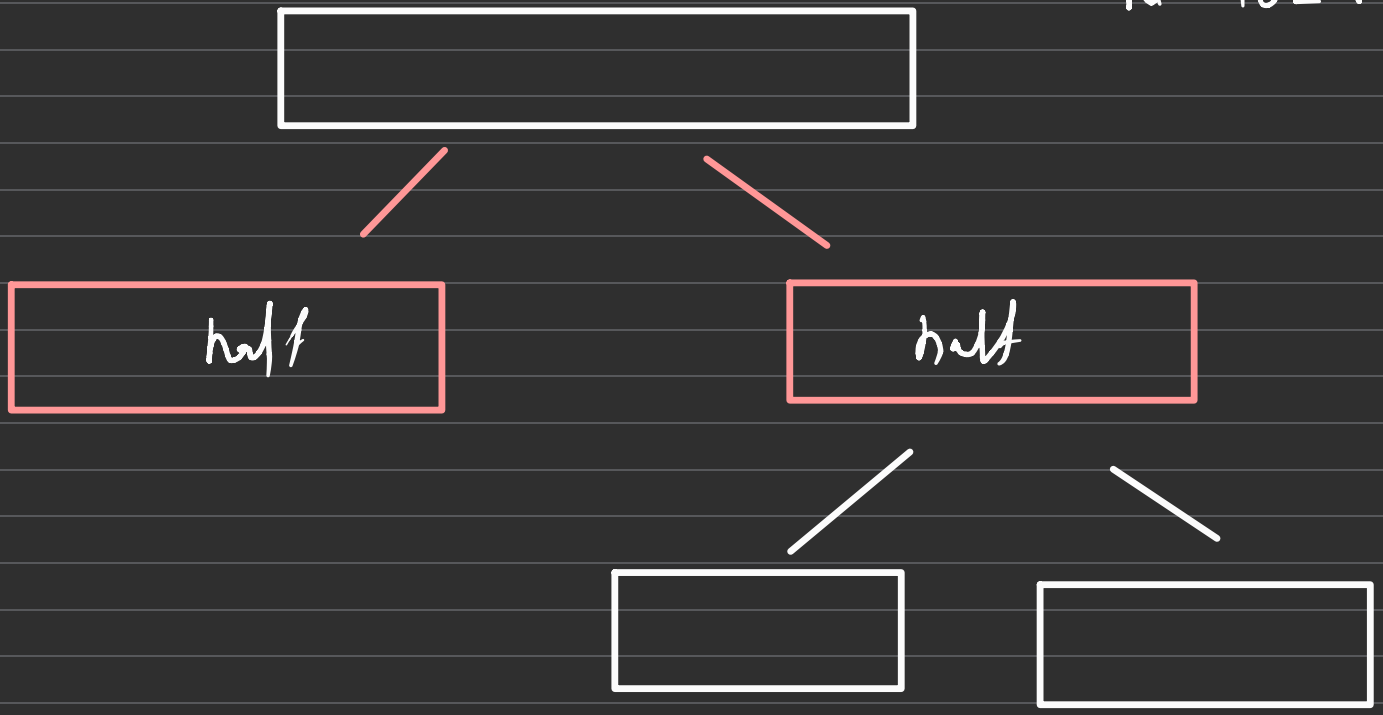
  s = m + 1
- if (a(m) > target) $a > b$

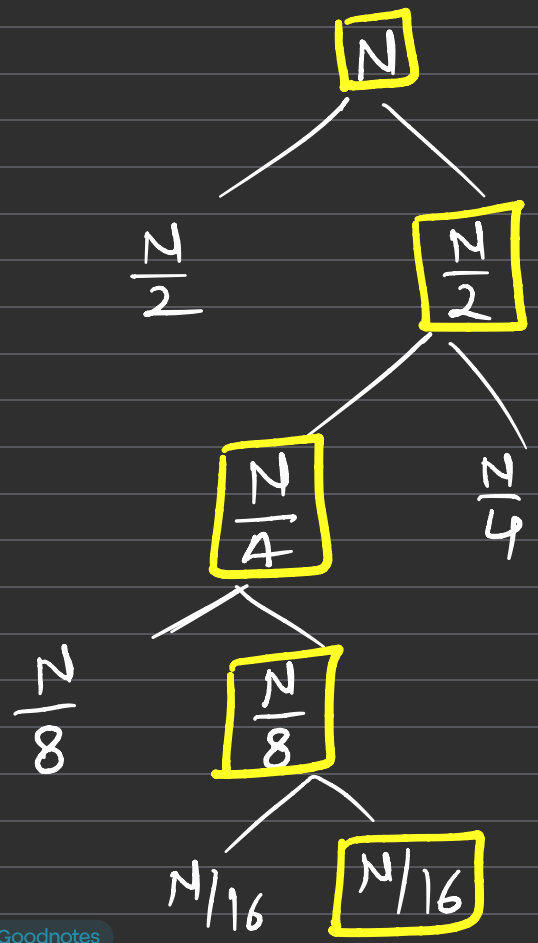
  e = m - 1

```
int binarySearch (arr[], N, target)
{
    int s=0, e= N-1
    while ( s<=e)
    {
        mid = (s+e)/2
        if (a(mid) == target)
            return index
        if (a(mid) < target)
            s = m+1
        else
            e = m-1
    }
    return -1
}
```

```cpp
int search(vector<int>& nums, int target) {

    int n = nums.size();

    // step 1: define the search space
    int start = 0, end = n - 1;

    while(start <= end){

        // step 2: find the mid
        int mid = (start + end) / 2;

        // a == b
        if (nums[mid] == target){
            return mid;
        }

        // a < b : move to right side
        if (nums[mid] < target){
            start = mid + 1;
        }

        // a > b : move to left
        else{
            end = mid - 1;
        }
    }

    return -1;
}
```

N = 1024





$N \rightarrow \frac{N}{2^0}$

$\frac{N}{2} \rightarrow \frac{N}{2^1}$

$\frac{N}{4} \rightarrow \frac{N}{2^2}$

$\frac{N}{8} \rightarrow \frac{N}{2^3}$

$\frac{N}{16} \rightarrow N/2^4$

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

$$N = 2^K$$

$$\log N = \log 2^K$$

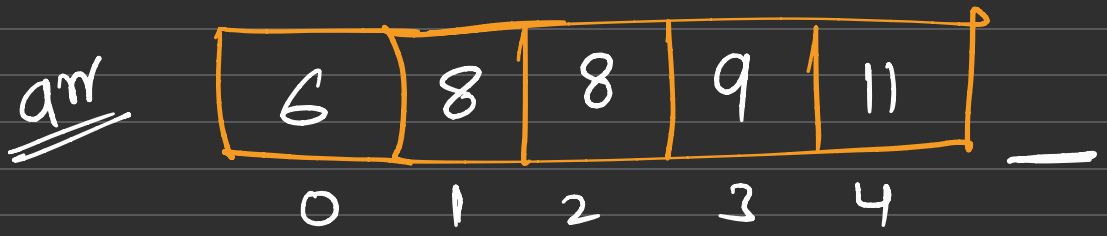
$$\log N = K \log 2$$

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

$$O(\log_2 N)$$

$$K = \log_2 N$$

Lower Bound



LB(8) = 1 (Return index if elem. present)

LB(7) = 1

LB(12) = N

Lower Bound

N = 4

x = 10

ans =

(c)

(s)

(m)

s = 0

e = 3

m = 1

s = 2

e = 3

M = 2

s = 3

e = 3

m = 3

s = 4

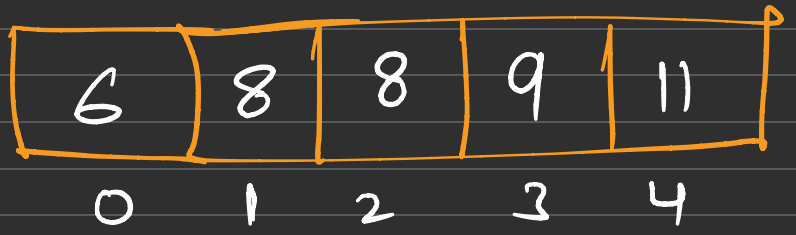
e = 3 ✗

3 ≥ 10

```
int LB (arr, N, x)
{
    ans = N
    s=0, e= N-1
    while (s ≤ e)
    {
        m = (s+e)/2
        if (a(m) ≥ x)
        {
            ans = a(mid)
            e = mid-1
        }
        else
            s = mid+1
    }
}
```

```cpp
int lowerBound(vector<int>& arr, int target) {

    int n = arr.size();

    int ans = n;
    int start = 0, end = n - 1;


    while(start <= end){

        // step 1: find the mid
        int mid = (start + end) / 2;

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

    return ans;
}
```

Upper Bound



N = 5

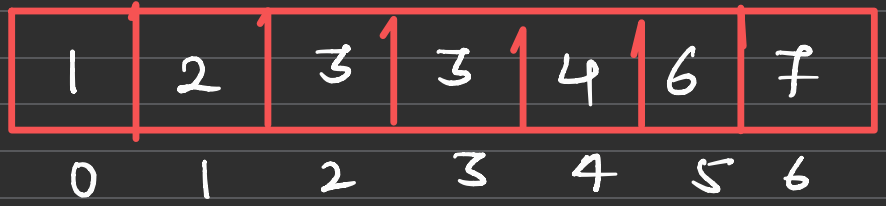
UB(6) = 1

UB(7) = 1

UB(12) = N

Same → LB

Upper Bound



N = 7

$x = 6$

LB

UB $>$ $x = 5$

```cpp
int upperBound(vector<int>& arr, int target) {
    int n = arr.size();

    int ans = n;
    int start = 0, end = n - 1;


    while(start <= end){

        // step 1: find the mid
        int mid = (start + end) / 2;

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

    return ans;
}
```