Newton School of Technology

5

Searching Algorithms Part-1

Introduction to Searching

Searching examples in real world ?

Searching examples in real world ?

- Google search
- YouTube video search
- Amazon product search
- Search for a file in system
- Search for a train in IRCTC app

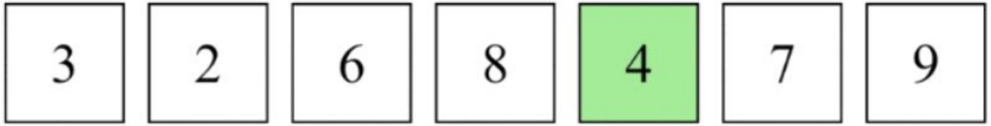
How will you find a specific book in a bookshelf ?

How will you find a specific book in a bookshelf ?

You search through the books one by one until you locate the desired one.

Find the correct Key!

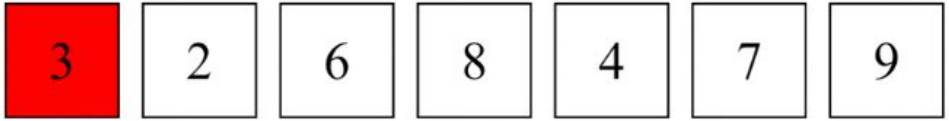
How do I search for an element in a List ?



Linear Search

Linear Search :

Search for key = 4 in given array.



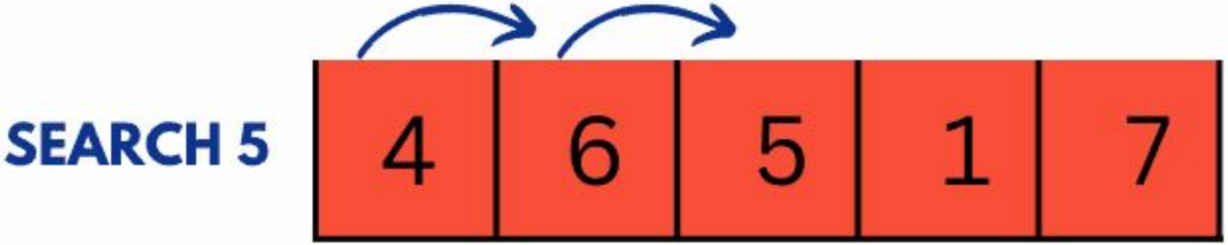
3	2	6	8	4	7	9

Linear Search:

- Algorithm for finding a specific value within a list.

- Works by sequentially checking each element of the list for the target value until a match is found or until all elements have been checked.

Linear Search:

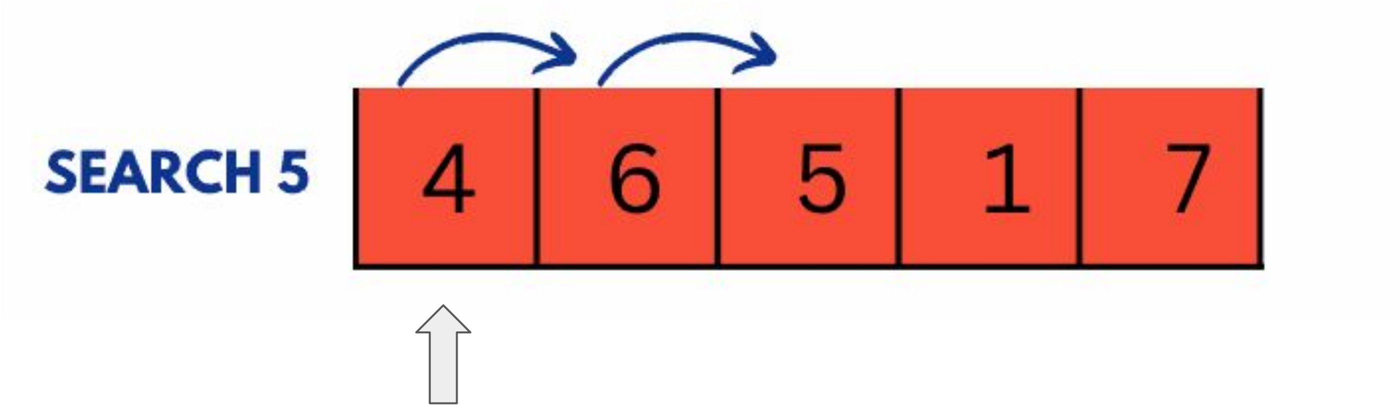


SEARCH 5

4	6	5	1	7

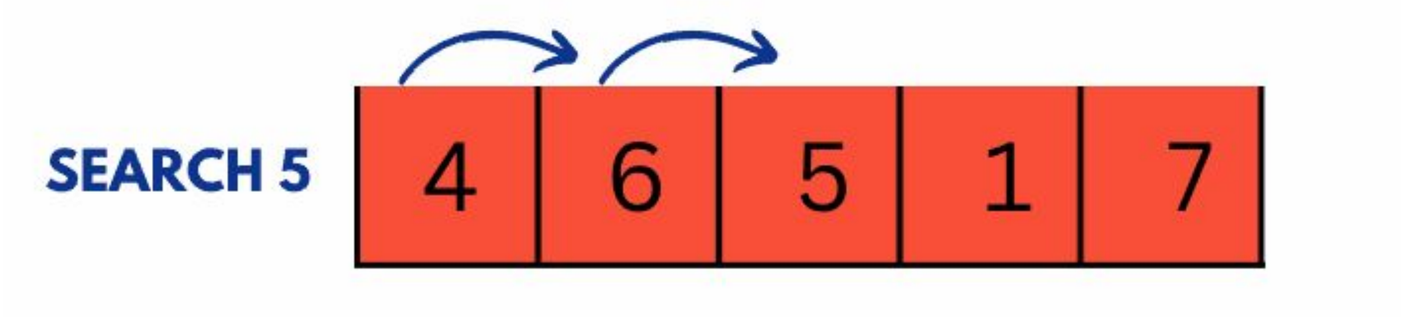
Steps for Linear Search :

1. **Start from the beginning**



Steps for Linear Search :

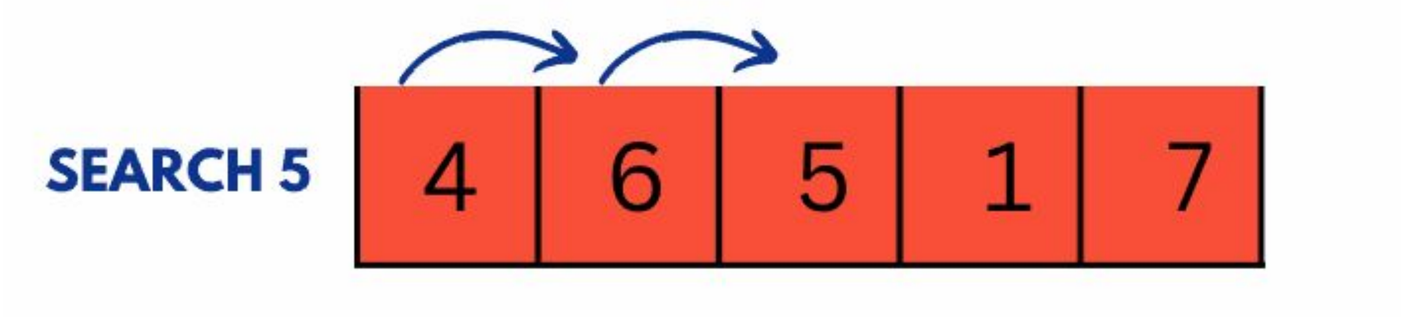
1. Start from the beginning
2. Iterate and check every element with target



4	6	5	1	7

Steps for Linear Search :

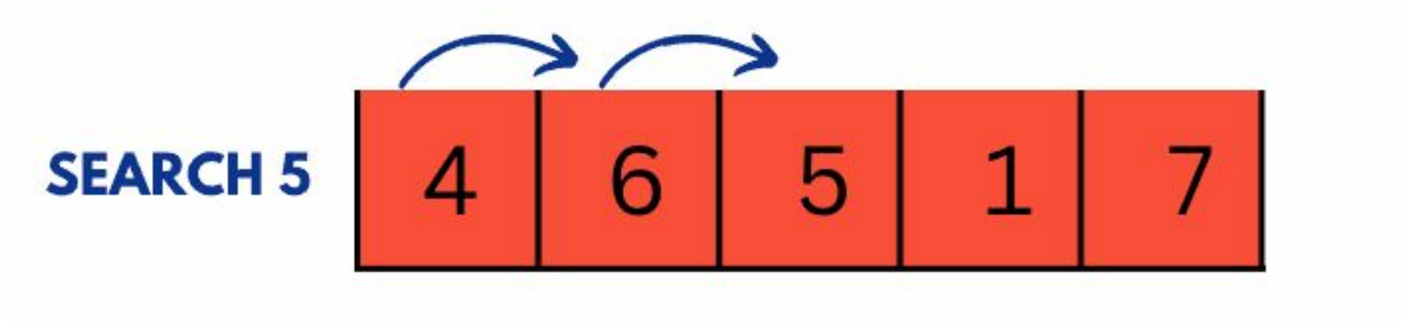
1. Start from the beginning
2. Iterate and check every element with target
3. Return index if element found



4	6	5	1	7

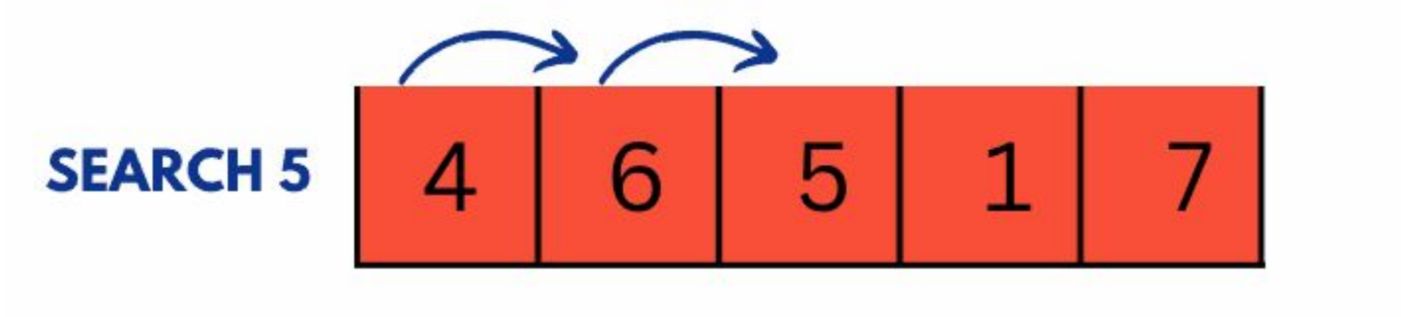
Steps for Linear Search :

1. Start from the beginning
2. Iterate and check every element with target
3. Return index if element found
4. Continue till end of the List



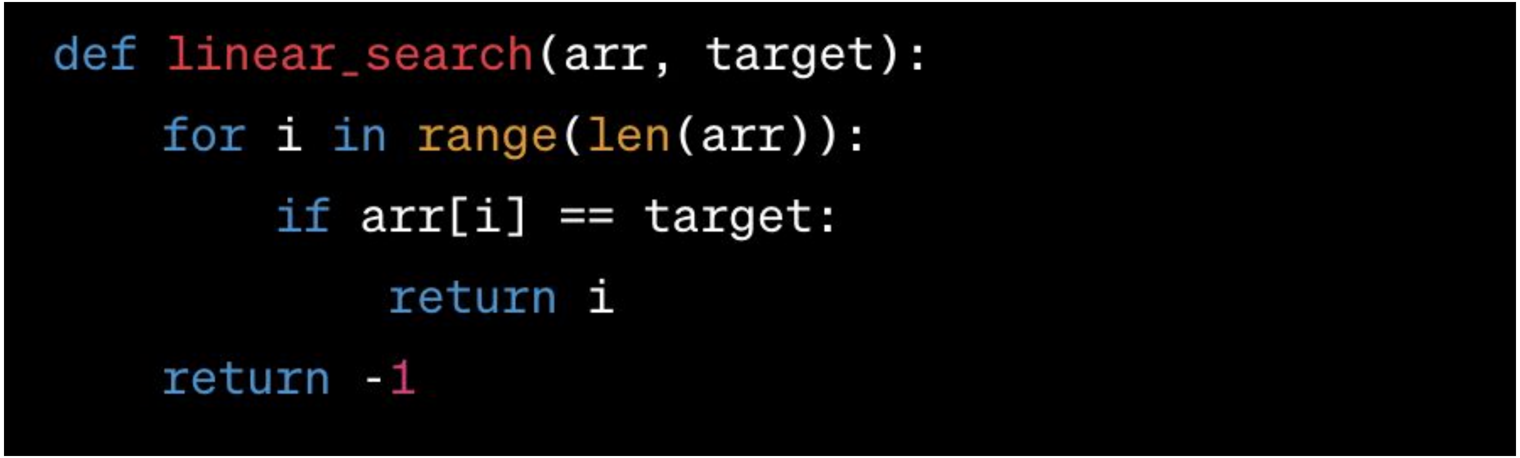
Steps for Linear Search :

1. Start from the beginning
2. Iterate and check every element with target
3. Return index if element found
4. Continue till end of the List
5. Return -1 if element not found



Visualizer Demo : Linear Search

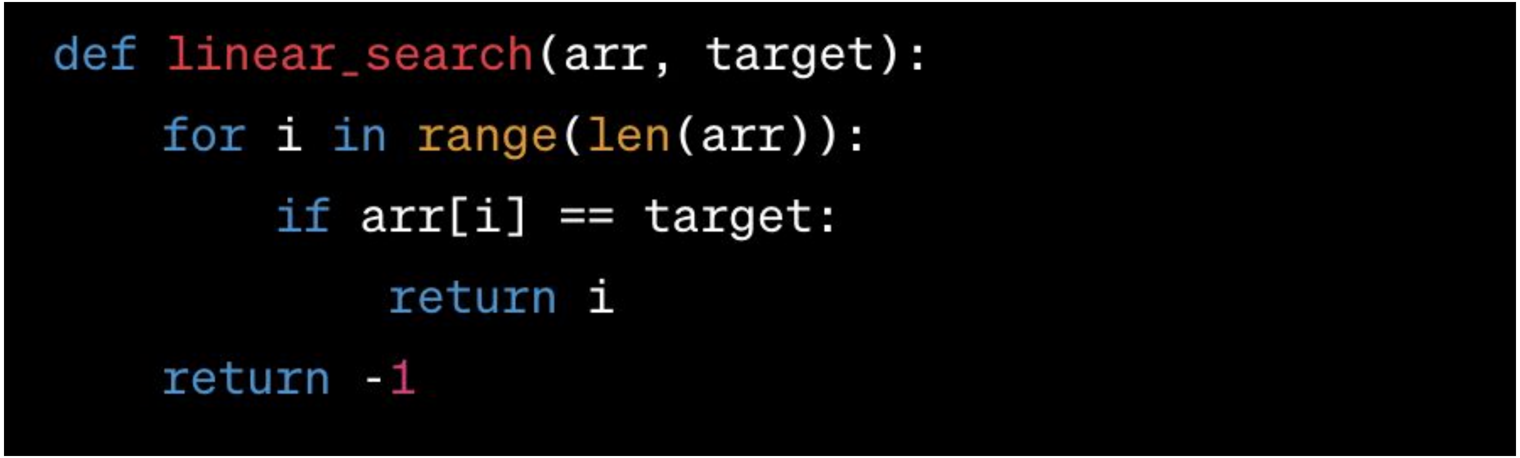
Implementation of Linear Search :



```python
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1
```

Time Complexity : ??

Implementation of Linear Search :



```python
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1
```

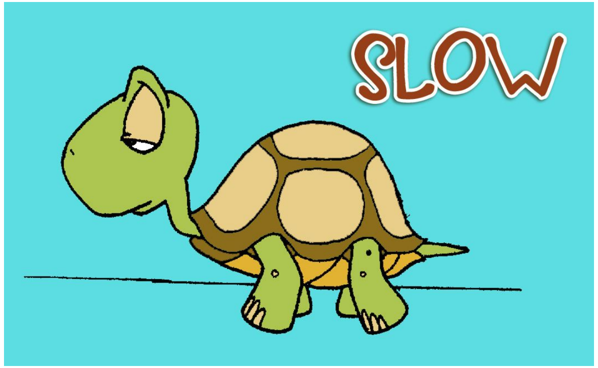
Time Complexity : O(n)

Q. Find Last Occurence of Character

Why Linear Search is considered Inefficient ?

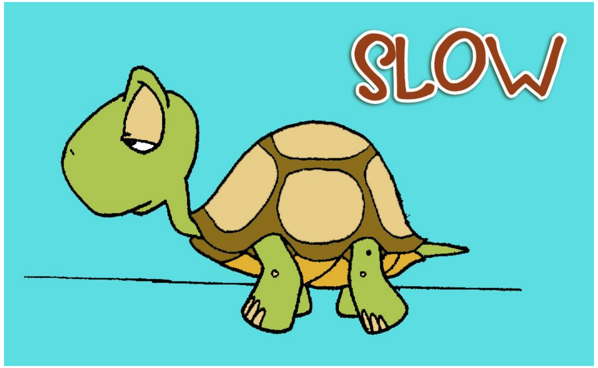
- In worst-case scenario , the algorithm needs to check every single element.

- As the size of the list grows, the time taken grows linearly, making it inefficient for large datasets.



Why Linear Search is considered Inefficient ?

Would you still search elements sequentially if the list was sorted ?



How do you search a word in a dictionary ?

How do you find your marks on grade sheet - sorted by names ?

Binary Search

Example - Target is present in the List

Suppose we have Sorted list of 16 elements

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Let search key value = k = 55

Suppose we have Sorted list of 16 elements

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0 mid=(0+15)/2=7 right = 15

Let search key value = k = 55

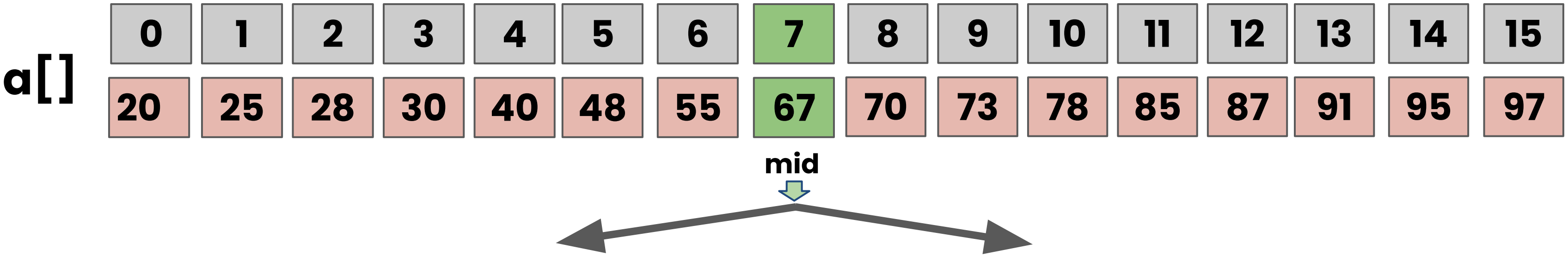
Suppose we have Sorted list of 16 elements

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0 mid=(0+15)/2=7 right = 15

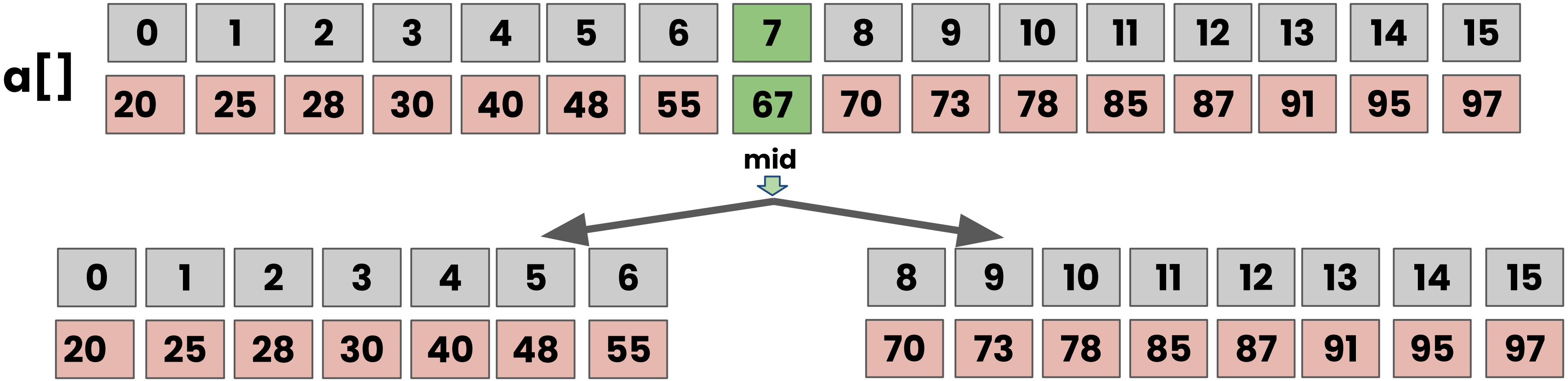
Let search key value = k = 55

key value to search (k) = 55



a[]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

key value to search (k) = 55



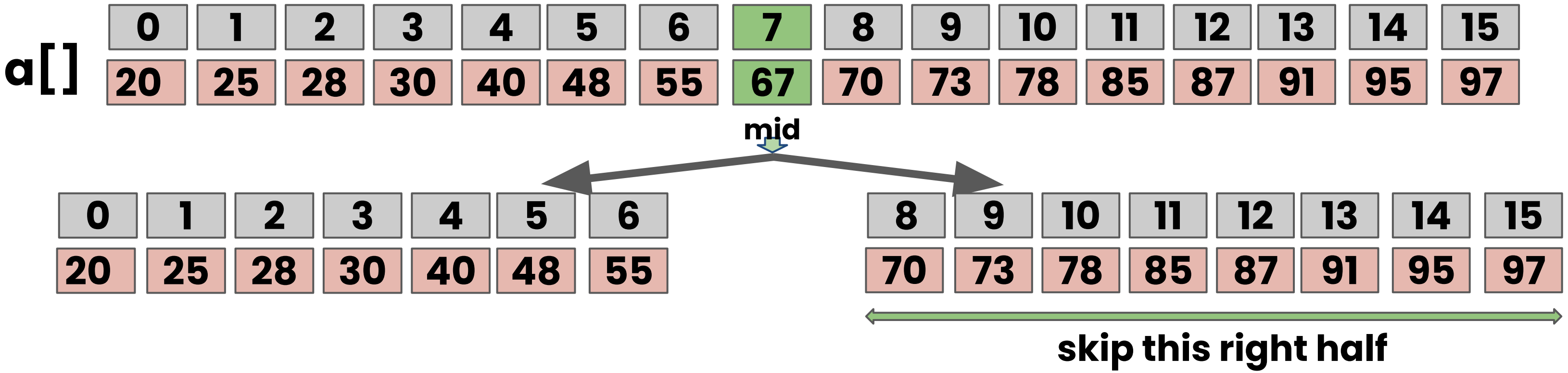
a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

0	1	2	3	4	5	6
20	25	28	30	40	48	55

8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

key value to search (k) = 55



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

0	1	2	3	4	5	6
20	25	28	30	40	48	55

8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

skip this right half

Case 1 : a[mid] = k Search is successful, return mid and exit

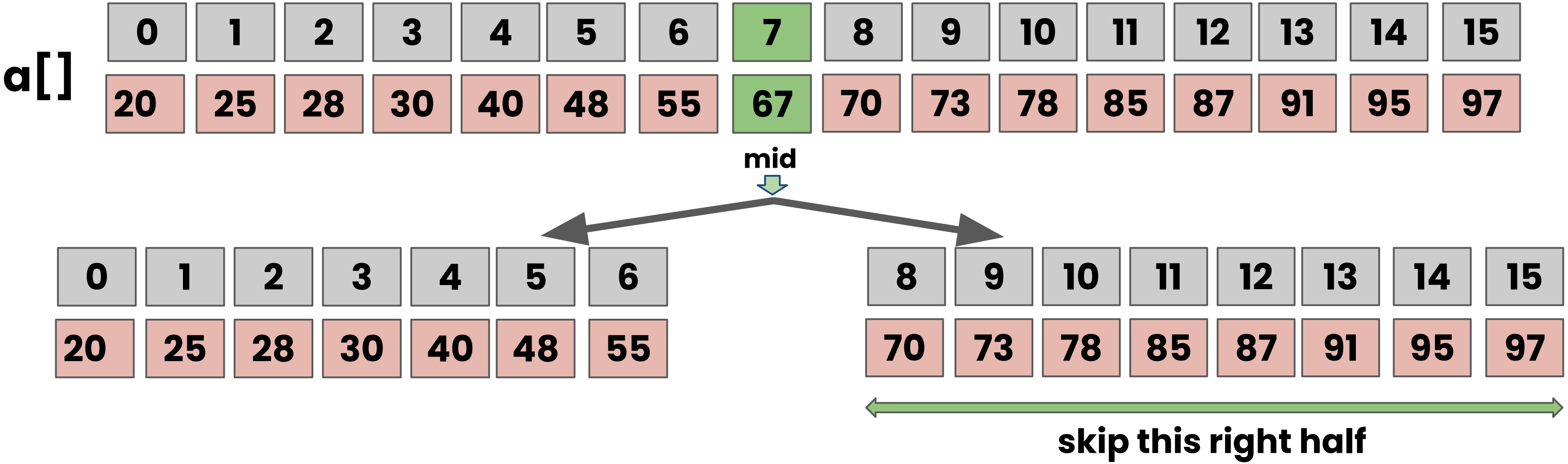
Case 2 : a[mid] > k Proceed your search only to the Left Half **Right = mid - 1**

Case 3: a[mid] < k Proceed your search only to the Right Half **Left = mid + 1**

Case 1 : a[7] = 55 ? ⇨ **false**

Case 2 : a[7] > 55 ⇨ **true** ⇨ **Proceed your search only to the Left Half**

Case 3: a[7] < 55 ⇨ **false**



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

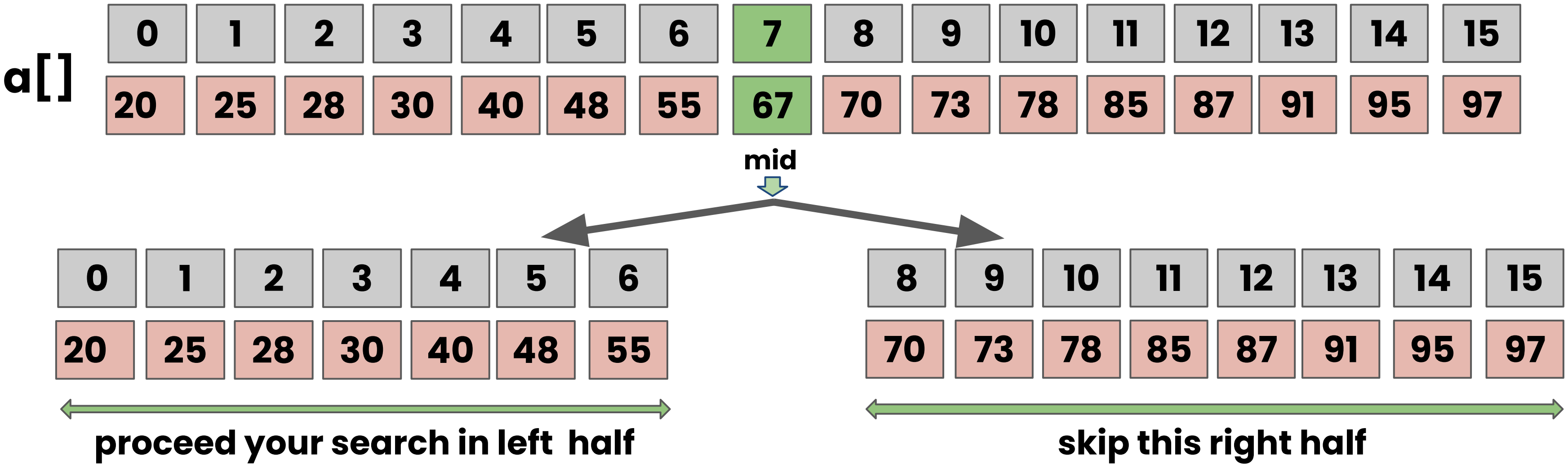
0	1	2	3	4	5	6
20	25	28	30	40	48	55

8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

skip this right half

key value to search (k) = 55

key value to search (k) = 55



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

0	1	2	3	4	5	6
20	25	28	30	40	48	55

proceed your search in left half

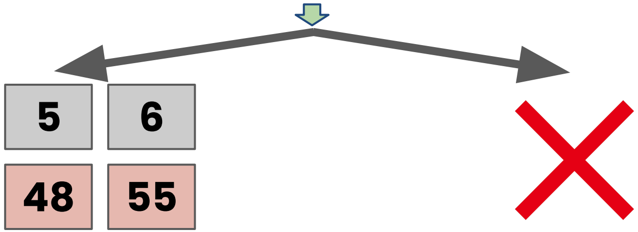
8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

skip this right half

key value to search (k) = 55

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

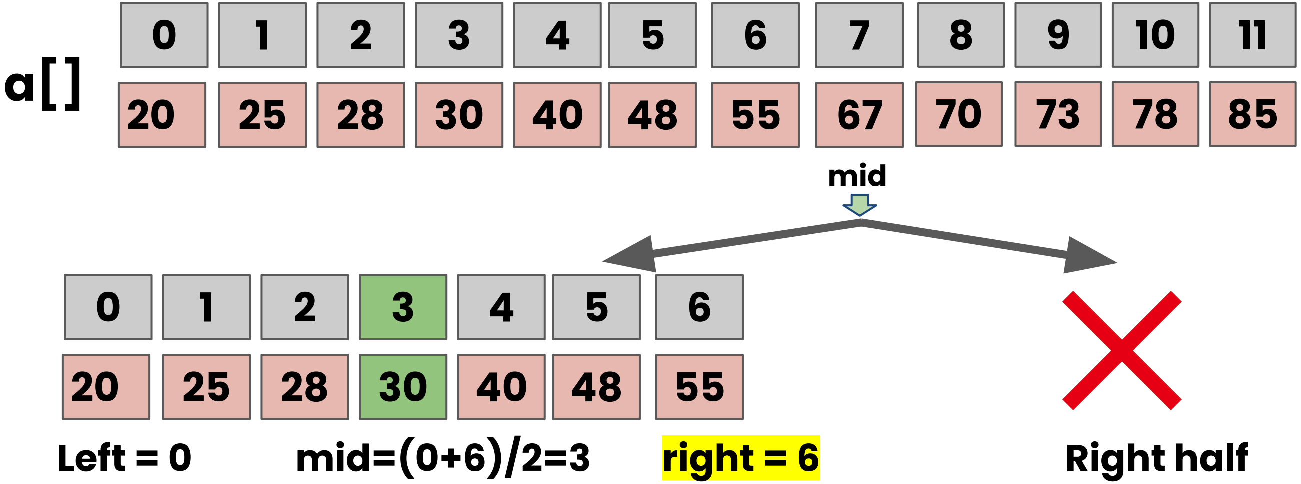


0	1	2	3	4	5	6
20	25	28	30	40	48	55

proceed your search in Left half

Right half

key value to search (k) = 55



a[]

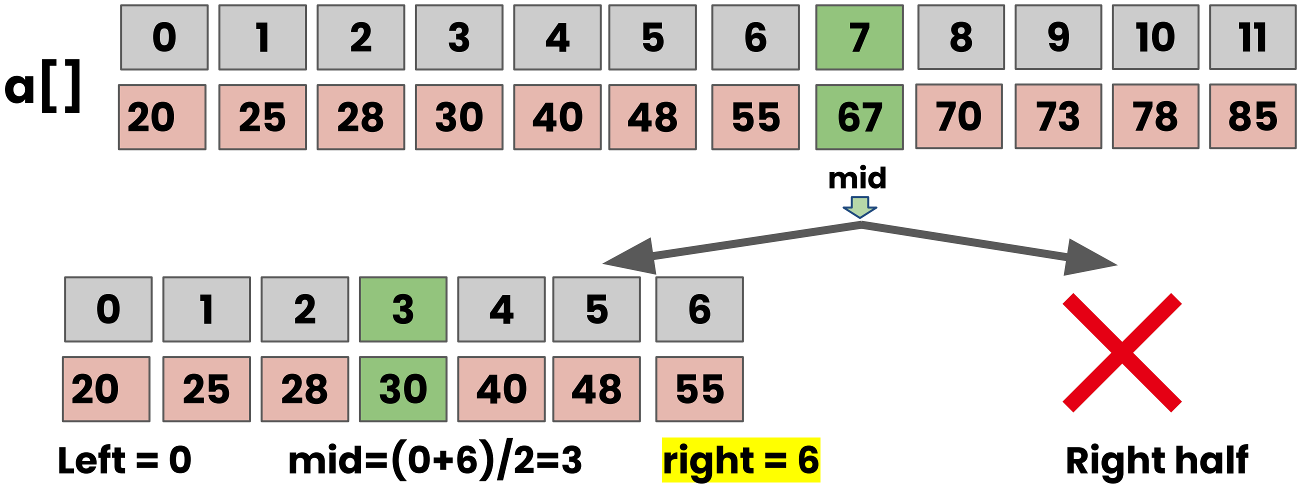
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

0	1	2	3	4	5	6
20	25	28	30	40	48	55

Left = 0 mid=(0+6)/2=3 right = 6

Right half

key value to search (k) = 55



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

0	1	2	3	4	5	6
20	25	28	30	40	48	55

Left = 0 mid=(0+6)/2=3 right = 6

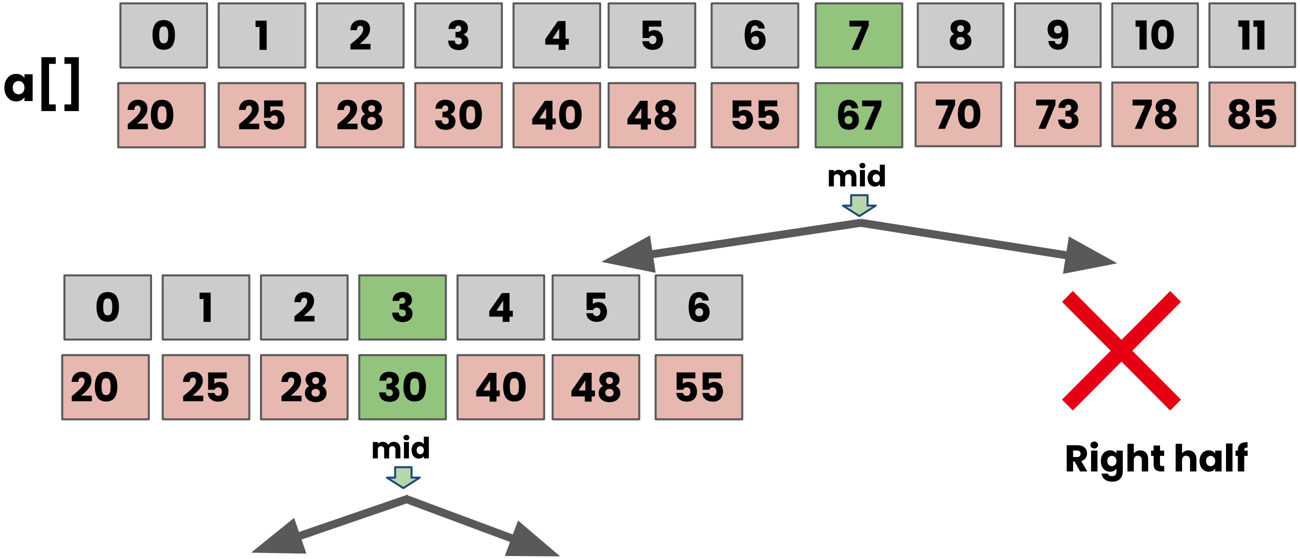
Right half

Case 1 : a[3] = 55 ? → false

Case 2 : a[3] > 55 ? → false

Case 3: a[3] < 55 ? → true → Proceed your search only to the Right Half

key value to search (k) = 55



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

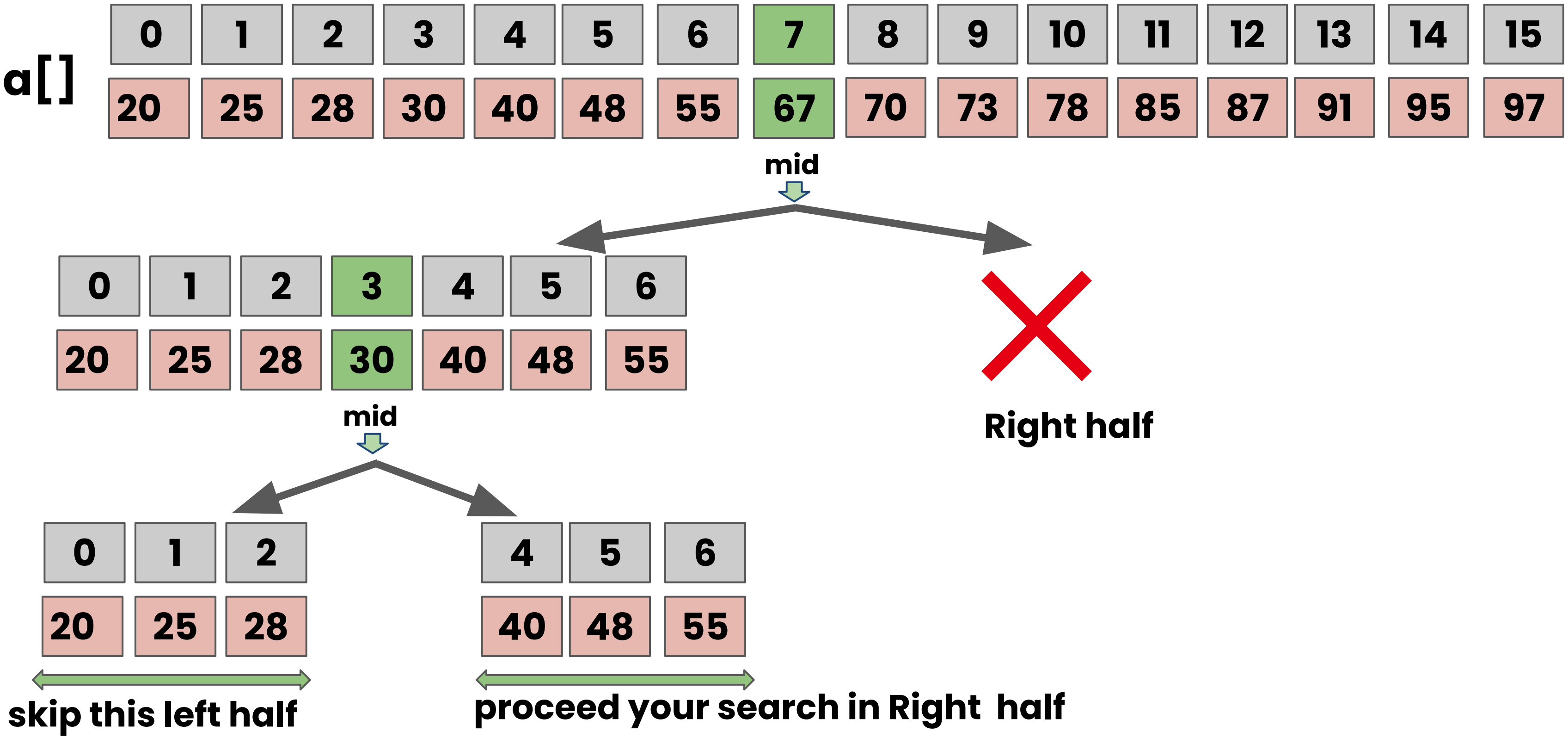
mid

0	1	2	3	4	5	6
20	25	28	30	40	48	55

mid

Right half

key value to search (k) = 55



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

0	1	2	3	4	5	6
20	25	28	30	40	48	55

Right half

mid

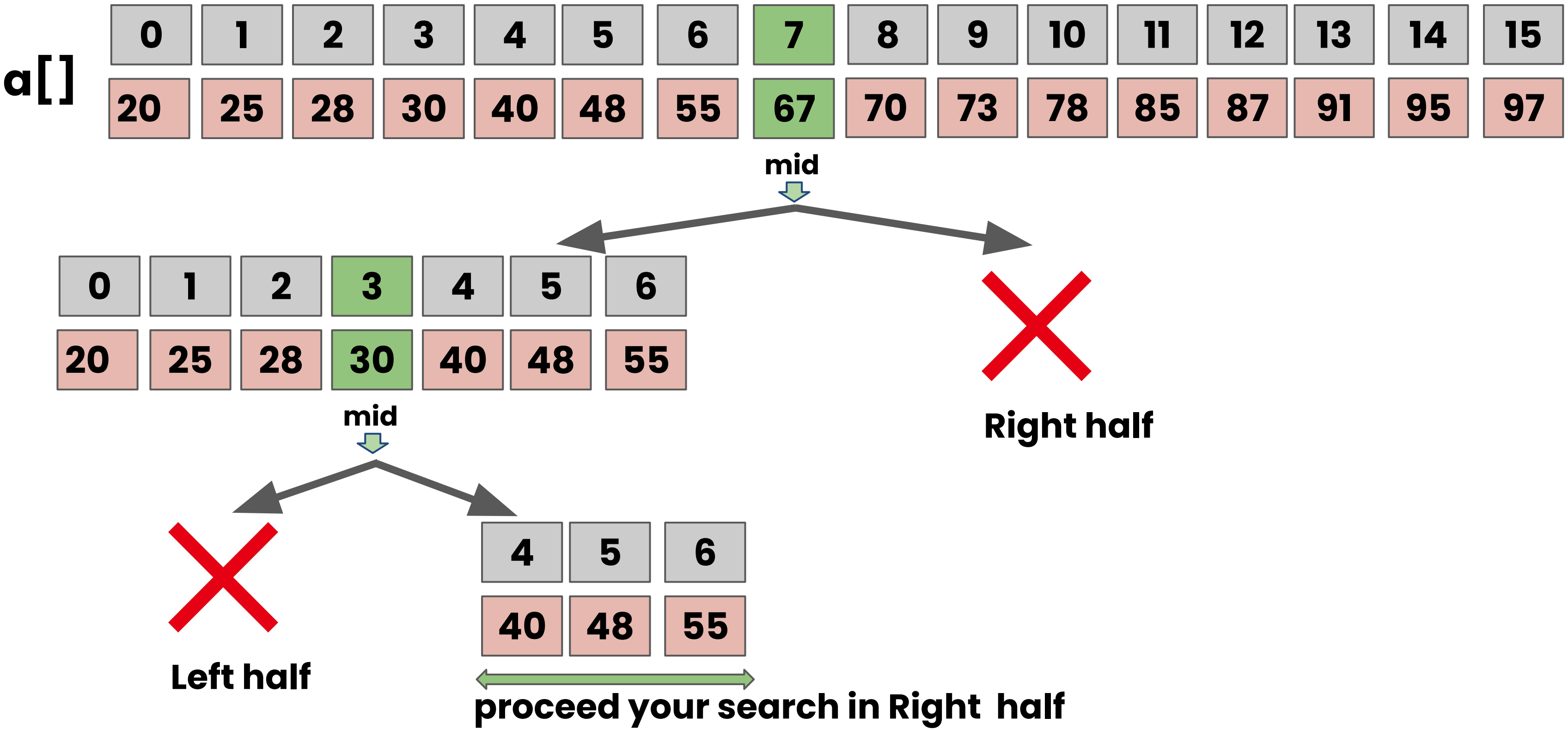
0	1	2
20	25	28

4	5	6
40	48	55

skip this left half

proceed your search in Right half

key value to search (k) = 55



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

0	1	2	3	4	5	6
20	25	28	30	40	48	55

mid

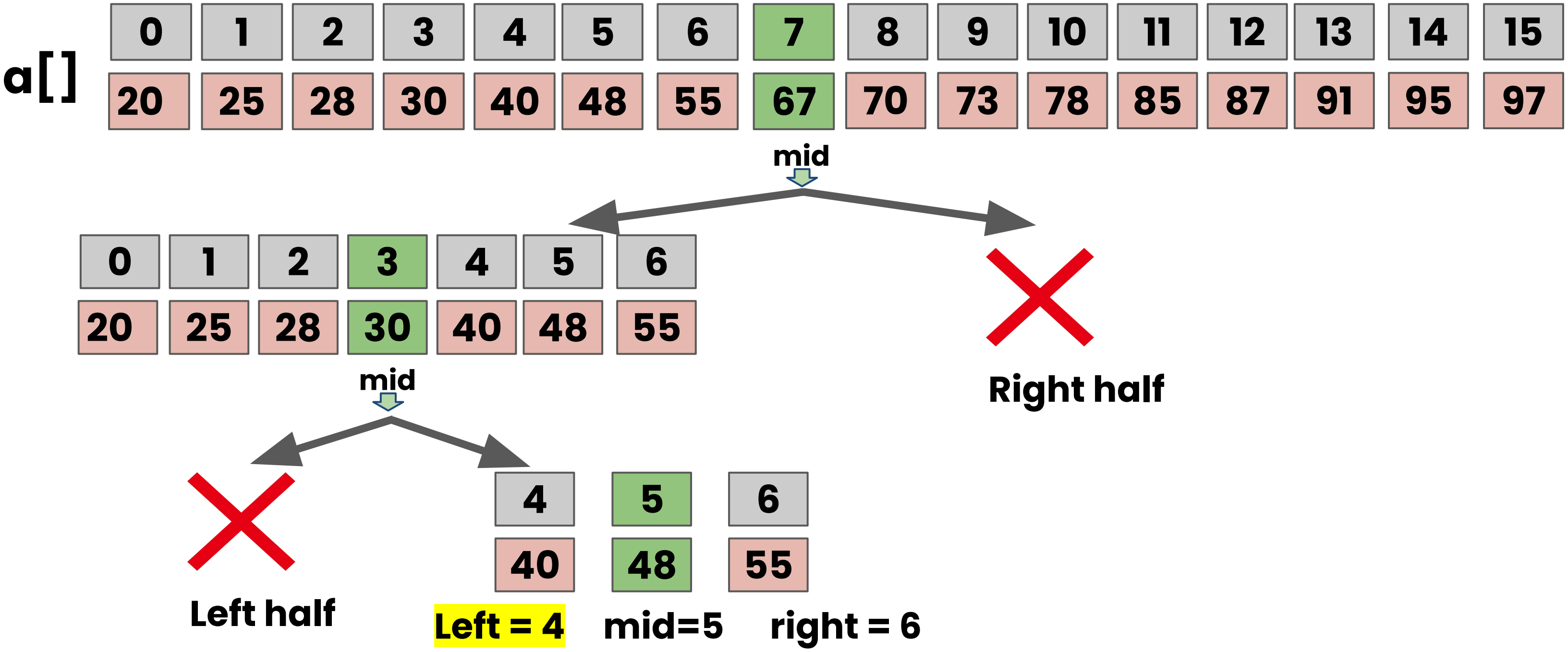
Right half

4	5	6
40	48	55

Left half

proceed your search in Right half

key value to search (k) = 55



a[]

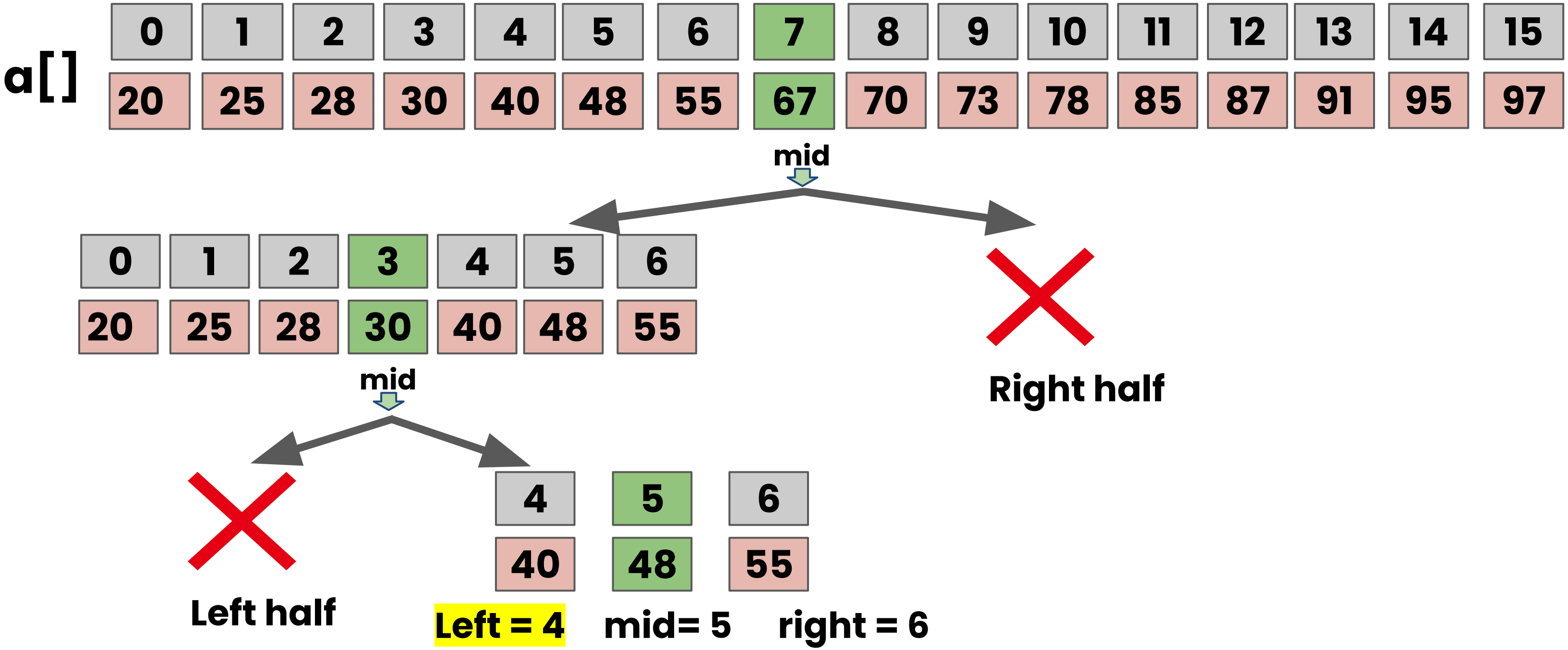
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

0	1	2	3	4	5	6
20	25	28	30	40	48	55

4	5	6
40	48	55

Left = 4 mid=5 right = 6

key value to search (k) = 55



a[]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

0	1	2	3	4	5	6
20	25	28	30	40	48	55

4	5	6
40	48	55

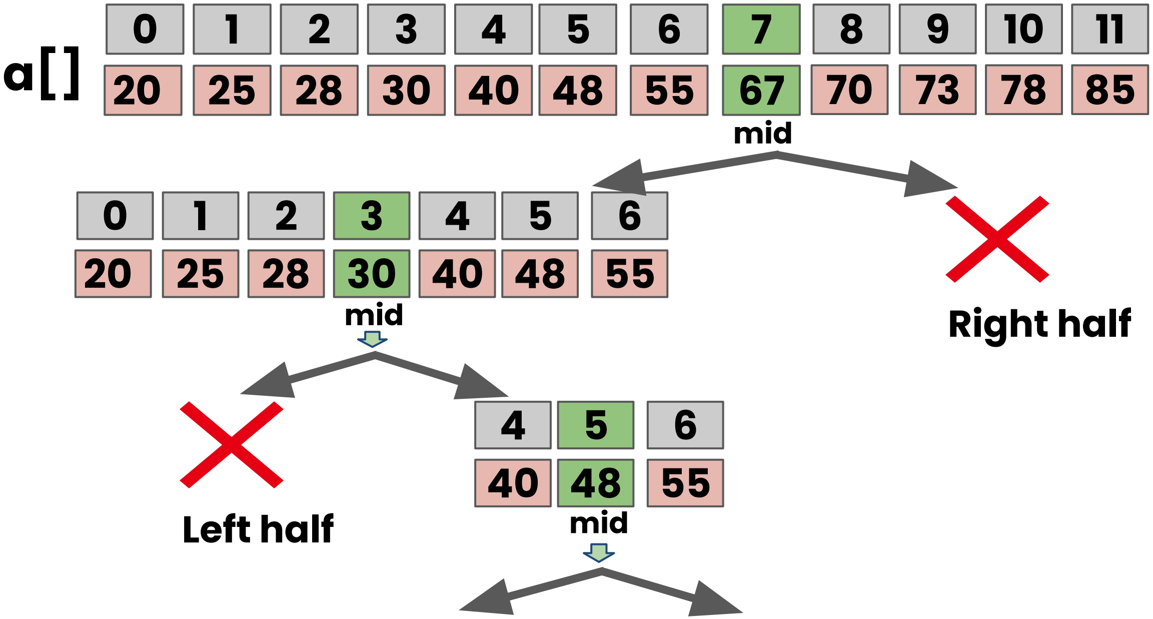
Left = 4 mid= 5 right = 6

Case 1 : a[5] = 55 ? ⇨ false

Case 2 : a[5] > 55 ? ⇨ false

Case 3: a[5] < 55 ? ⇨ true ⇨ Proceed your search only to the Right Half

key value to search (k) = 55



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

0	1	2	3	4	5	6
20	25	28	30	40	48	55

mid

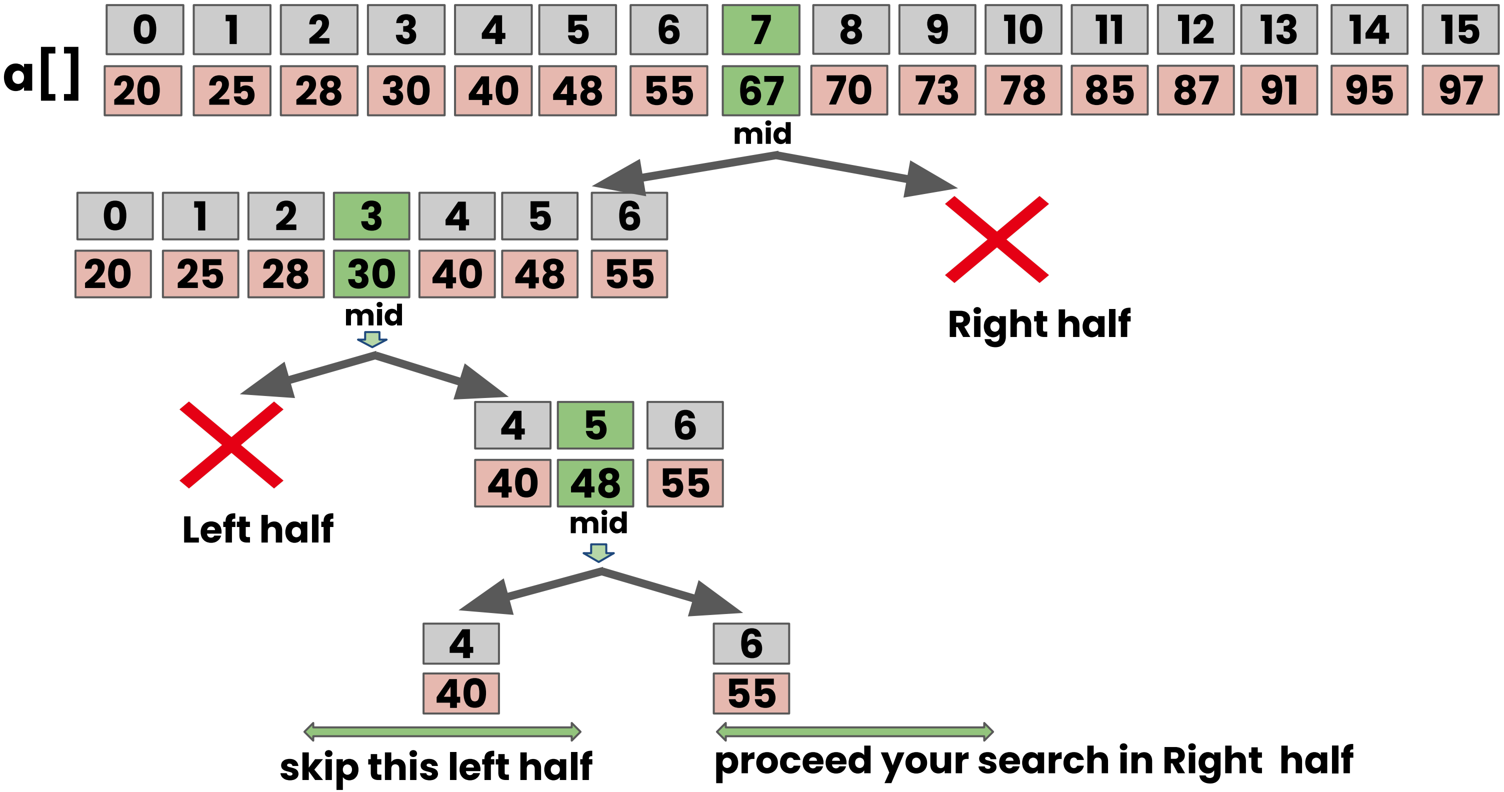
Right half

Left half

4	5	6
40	48	55

mid

key value to search (k) = 55



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

0	1	2	3	4	5	6
20	25	28	30	40	48	55

mid

Right half

Left half

4	5	6
40	48	55

mid

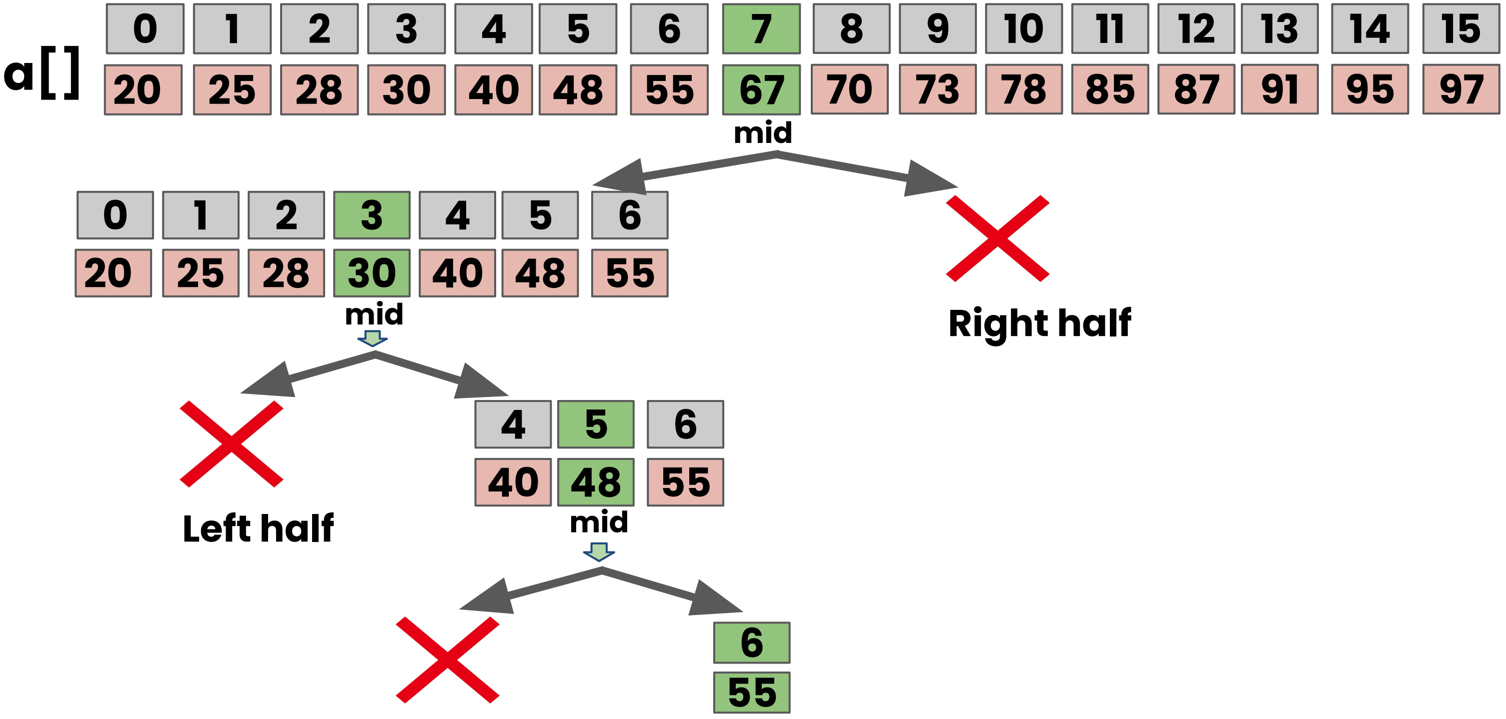
4
40

6
55

skip this left half

proceed your search in Right half

key value to search (k) = 55



a[]

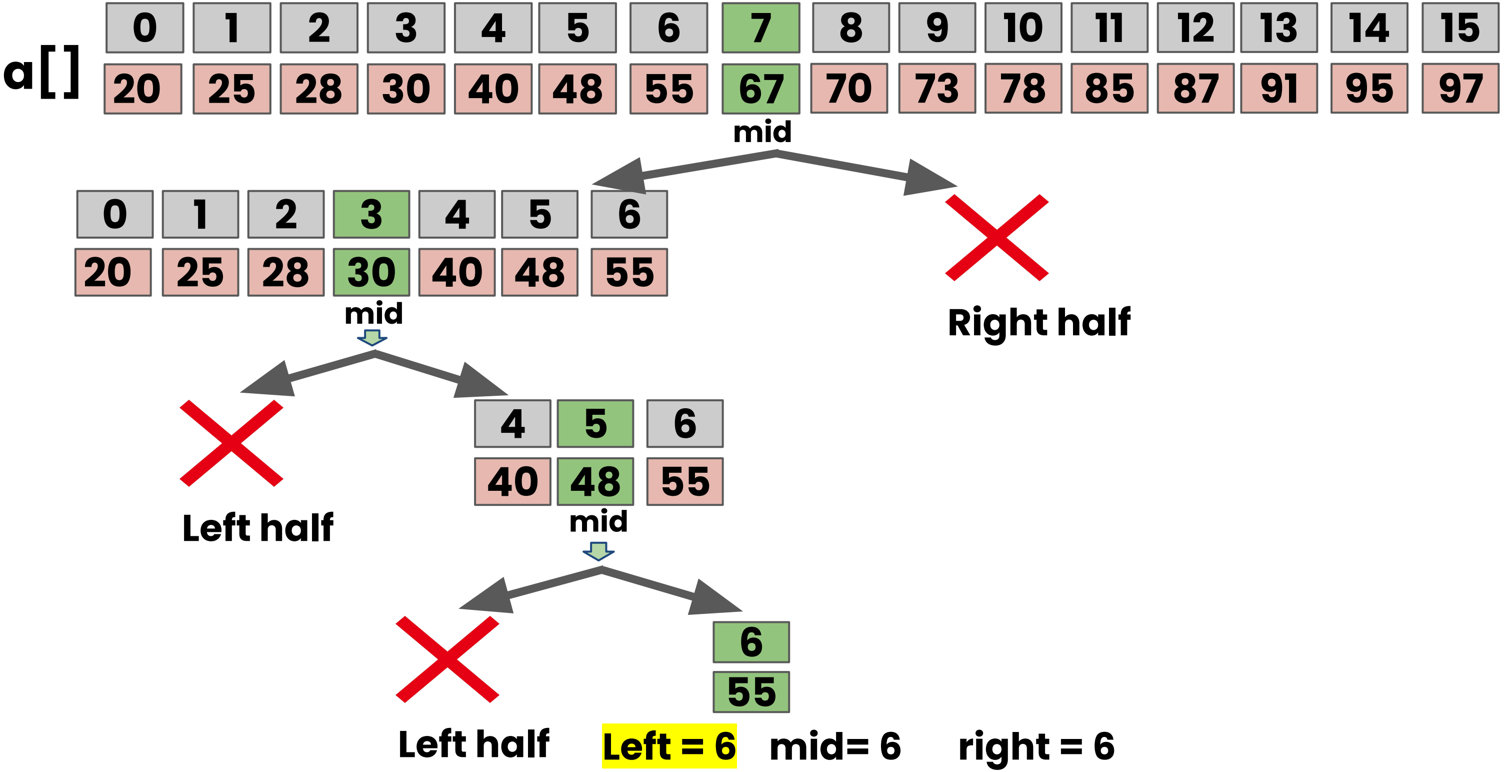
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

0	1	2	3	4	5	6
20	25	28	30	40	48	55

4	5	6
40	48	55

6
55

Left half **Left = 6** mid= 6 right = 6



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

0	1	2	3	4	5	6
20	25	28	30	40	48	55

mid

Right half

4	5	6
40	48	55

mid

Left half

6
55

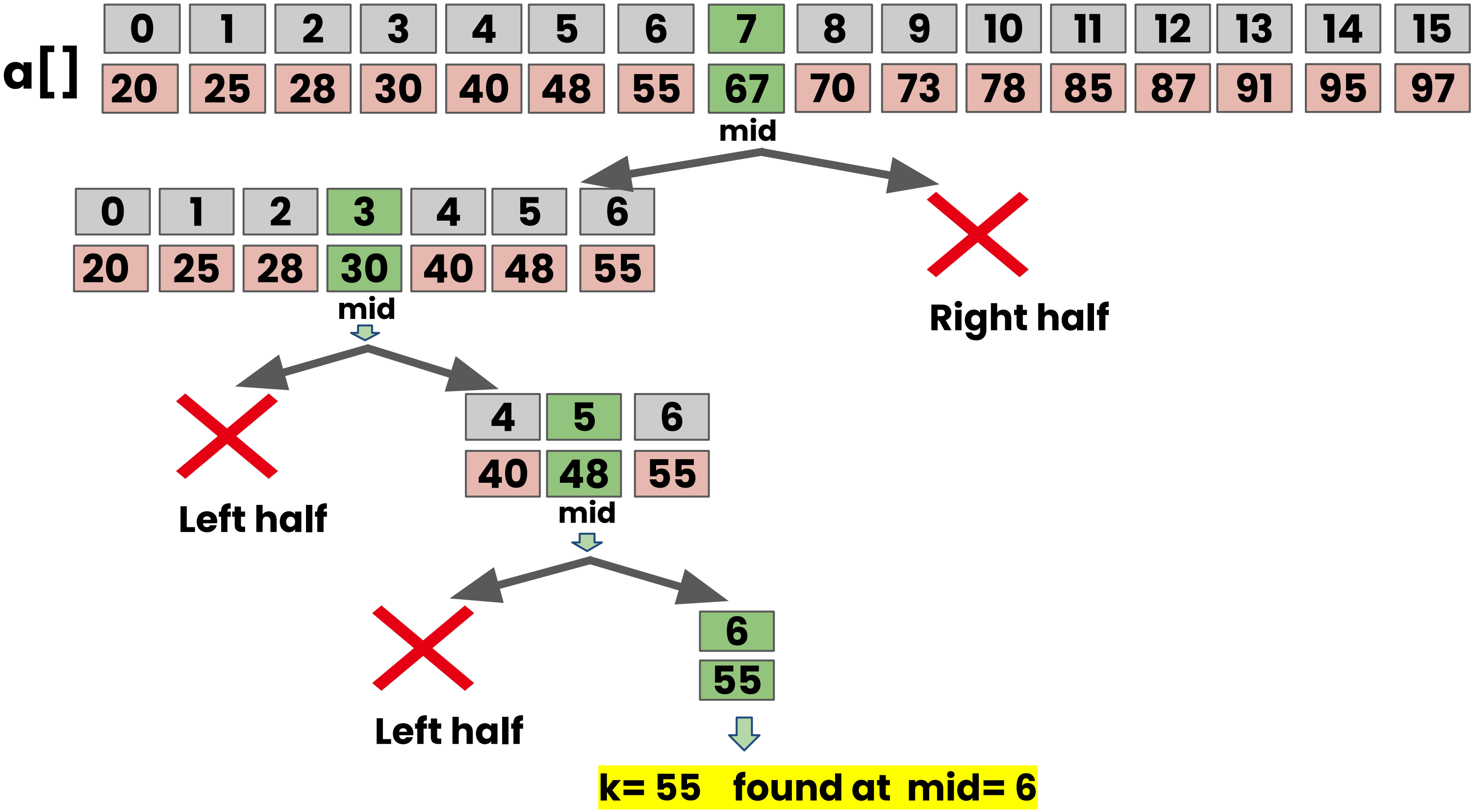
Left half

Left = 6 mid= 6 right = 6

Case 1 : a[6] = 55 ? ⇨ true ⇨ Search is successful, return mid = 6 and exit

Case 2 : a[6] > 55 ? ⇨ false

Case 3: a[6] < 55 ? ⇨ false



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

mid

Right half

0	1	2	3	4	5	6
20	25	28	30	40	48	55

mid

Left half

4	5	6
40	48	55

mid

Left half

6
55

k= 55 found at mid= 6

Key Element is not present in the List

Suppose we have Sorted list of 16 elements

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Suppose we have Sorted list of 16 elements

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0 **mid=(0+15)/2=7** **right = 15**

Let search key value = k = 96

Suppose we have Sorted list of 16 elements

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0 **mid=(0+15)/2=7** **right = 15**

Let search key value = k = 96

Case 1 : a[mid] = k Search is successful, return mid and exit

Case 2 : a[mid] > k Proceed your search only to the Left Half ⇨ **Right = mid – 1**

Case 3: a[mid] < k Proceed your search only to the Right Half ⇨ **Left = mid + 1**

Suppose we have Sorted list of 16 elements

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0 **mid=(0+15)/2=7** **right = 15**

Let search key value = k = 96

Case 1 : a[mid] = k Search is successful, return mid and exit

Case 2 : a[mid] > k Proceed your search only to the Left Half → **Right = mid - 1**

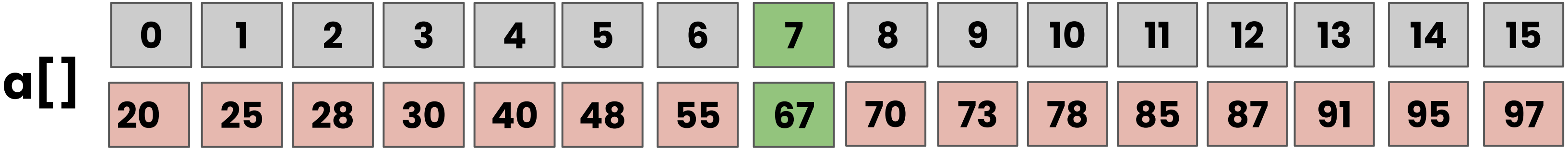
Case 3: a[mid] < k Proceed your search only to the Right Half → **Left = mid + 1**

Case 1 : a[7] = 96 ? → **false**

Case 2 : a[7] > 96 → **false**

Case 3: a[7] < 96 → **true** → **Proceed your search only to the Right Half**

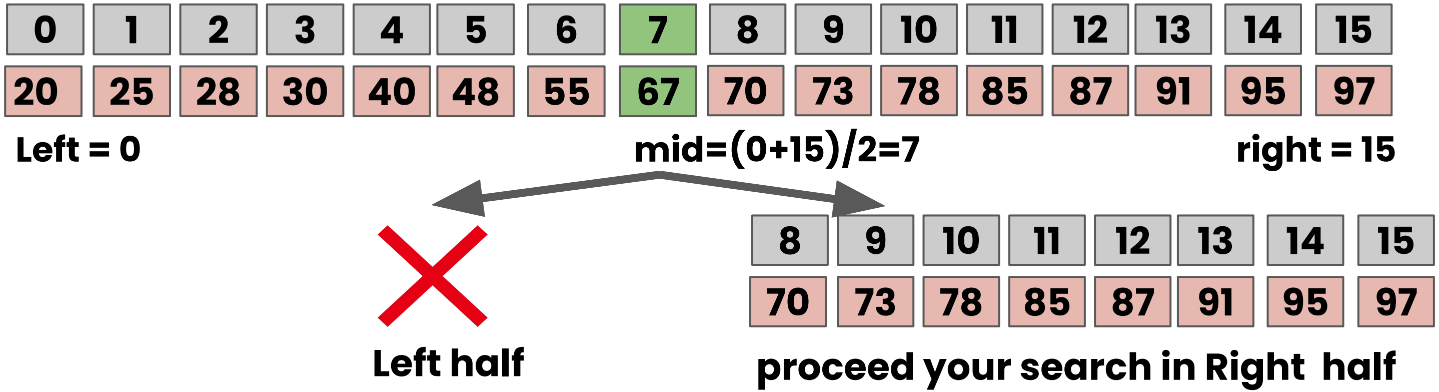
Let search key value (k) = 96



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[]	20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0 mid=(0+15)/2=7 right = 15

Let search key value (k) = 96



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0

mid=(0+15)/2=7

right = 15

Left half

8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

proceed your search in Right half

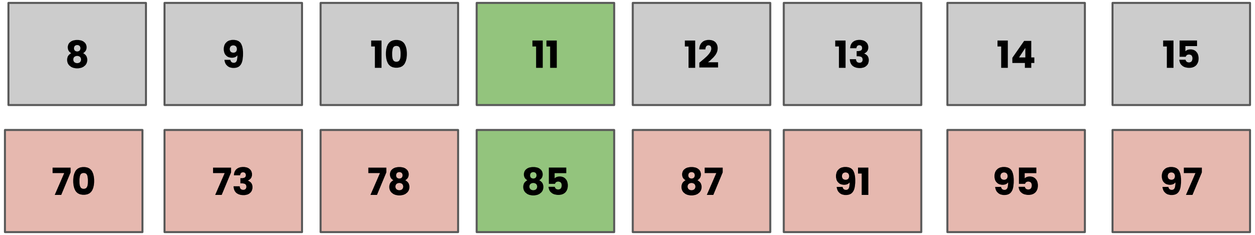
Let search key value (k) = 96

8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

Left = 8

right = 15

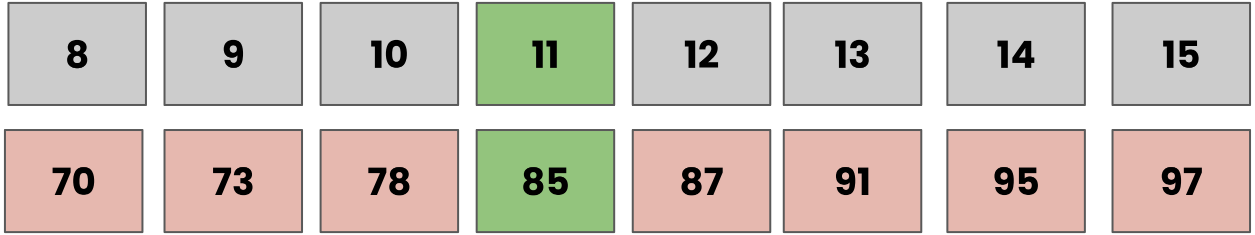
Let search key value (k) = 96



8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

Left = 8 mid=(8+15)/2=11 right = 15

Let search key value (k) = 96



8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

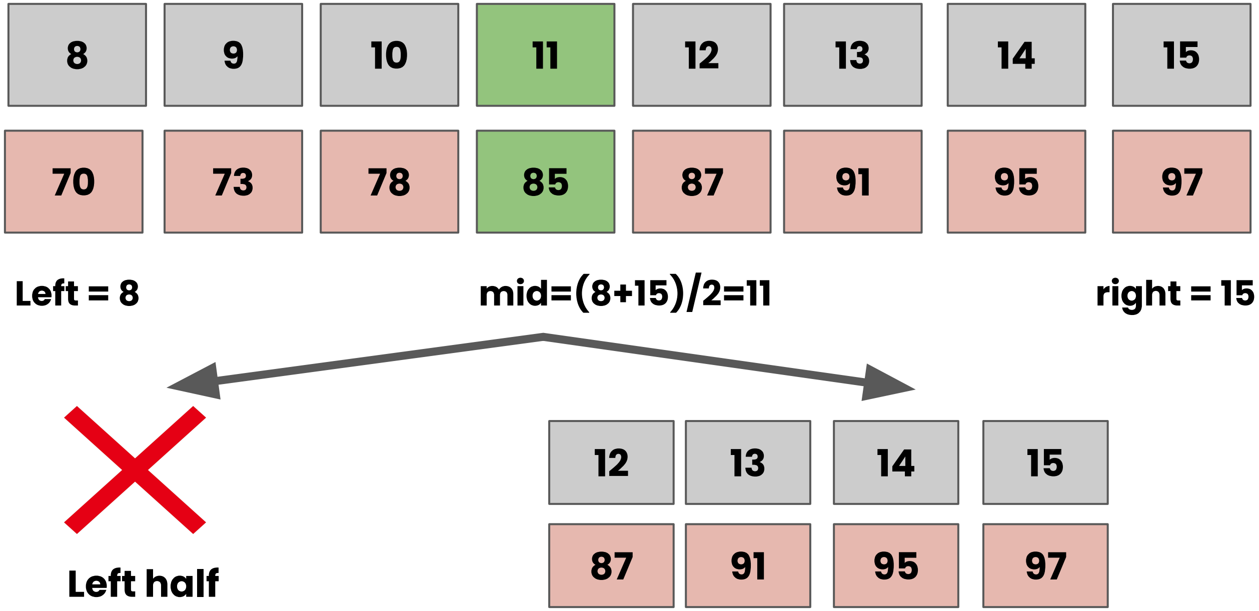
Left = 8 mid=(8+15)/2=11 right = 15

Case 1 : a[11] = 96 ? ⇨ false

Case 2 : a[11] > 96 ? ⇨ false

Case 3: a[11] < 96 ? ⇨ true ⇨ Proceed your search only to the Right Half

Let search key value (k) = 96



8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

Left = 8

mid=(8+15)/2=11

right = 15

Left half

12	13	14	15
87	91	95	97

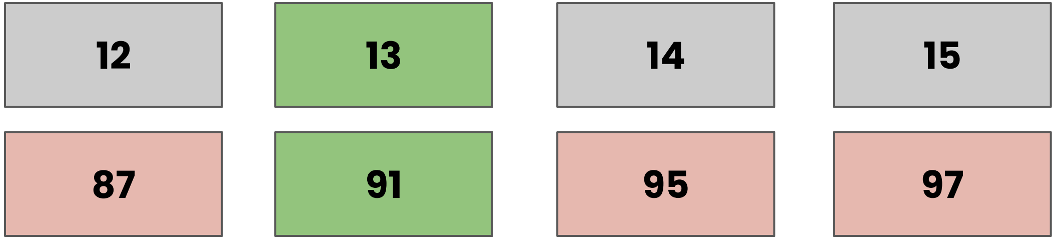
proceed your search in Right half

Let search key value (k) = 96

12	13	14	15
87	91	95	97

Left = 12
right = 15

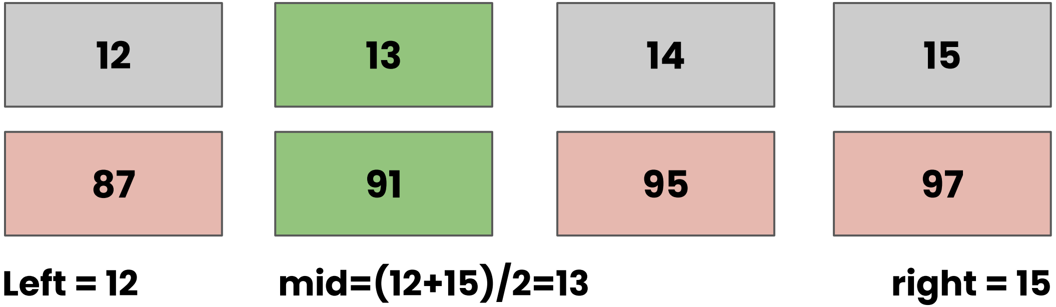
Let search key value (k) = 96



12	13	14	15
87	91	95	97

Left = 12 mid=(12+15)/2=13 right = 15

Let search key value (k) = 96



12	13	14	15
87	91	95	97

Left = 12 mid=(12+15)/2=13 right = 15

Case 1 : a[13] = 96 ? → false

Case 2 : a[13] > 96 ? → false

Case 3: a[13] < 96 ? → true → Proceed your search only to the Right Half

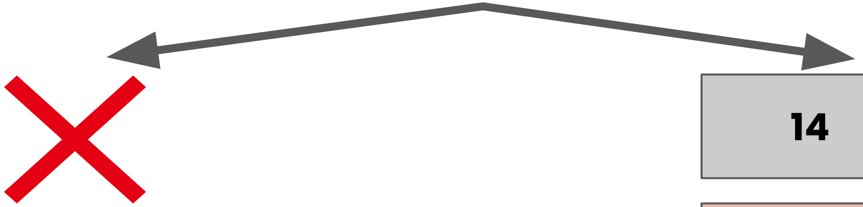
Let search key value (k) = 96

12	13	14	15
87	91	95	97

Left = 12

mid=(12+15)/2=13

right = 15

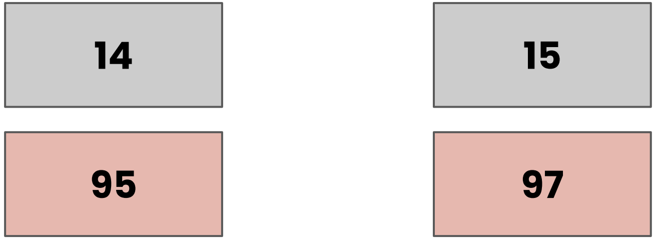


Left half

14	15
95	97

proceed your search in Right half

Let search key value (k) = 96



14

95

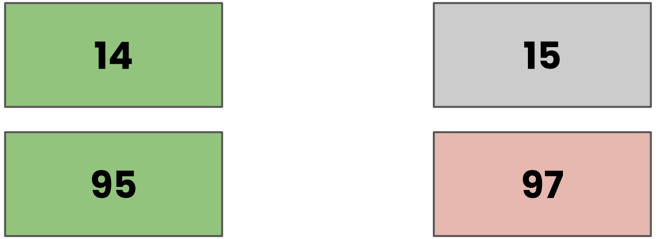
15

97

Left = 14

right = 15

Let search key value (k) = 96



14	15
95	97

Left = 14 right = 15

mid=(14+15)/2=14

Let search key value (k) = 96

14	15
95	97

Left = 14 right = 15

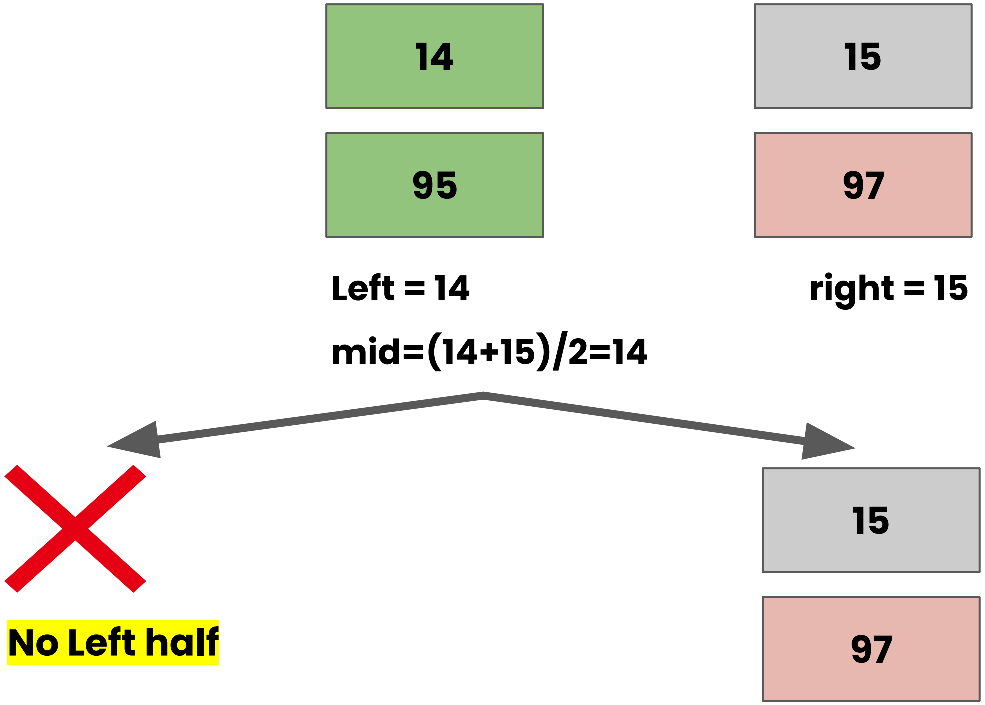
mid=(14+15)/2=14

Case 1 : a[14] = 96 ? → false

Case 2 : a[14] > 96 ? → false

Case 3: a[14] < 96 ? → true → Proceed your search only to the Right Half

Let search key value (k) = 96



14

15

95

97

Left = 14

right = 15

mid=(14+15)/2=14

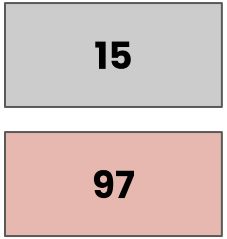
No Left half

15

97

proceed your search in Right half

Let search key value (k) = 96

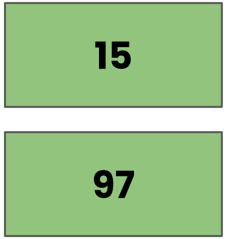


15

97

Left = 15 right = 15

Let search key value (k) = 96

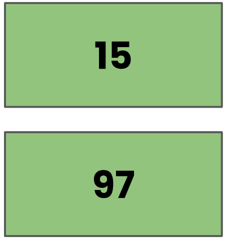


15

97

Left = 15 mid=(15+15)/2=15 right = 15

Let search key value (k) = 96



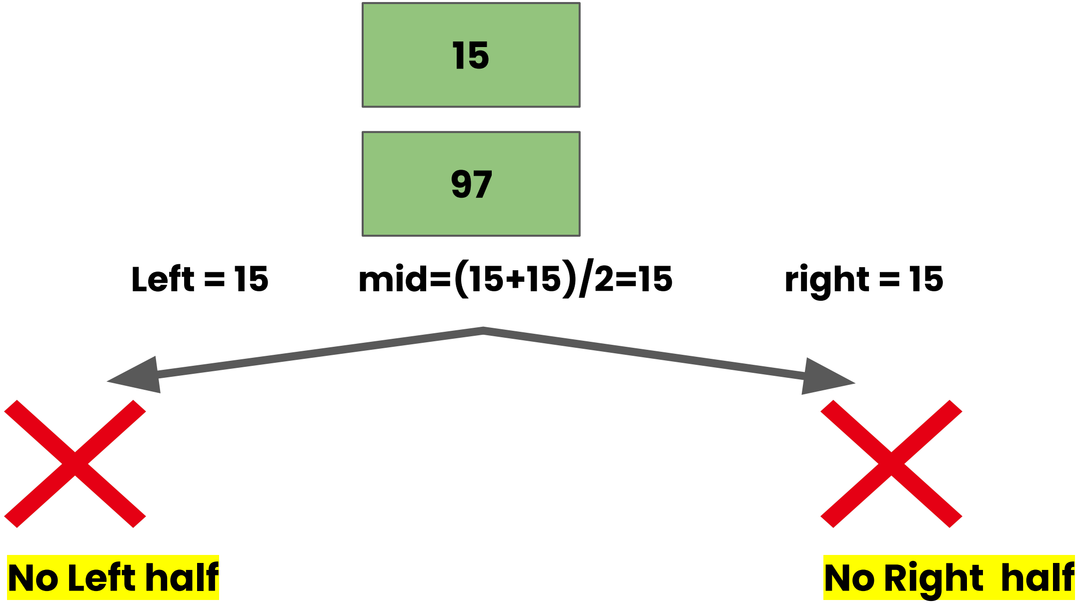
Left = 15 mid=(15+15)/2=15 right = 15

Case 1 : a[15] = 96 ? → false

Case 2 : a[15] > 96 ? → false → Proceed your search only to the Left Half

Case 3: a[15] < 96 ? → false

Let search key value (k) = 96



15

97

Left = 15

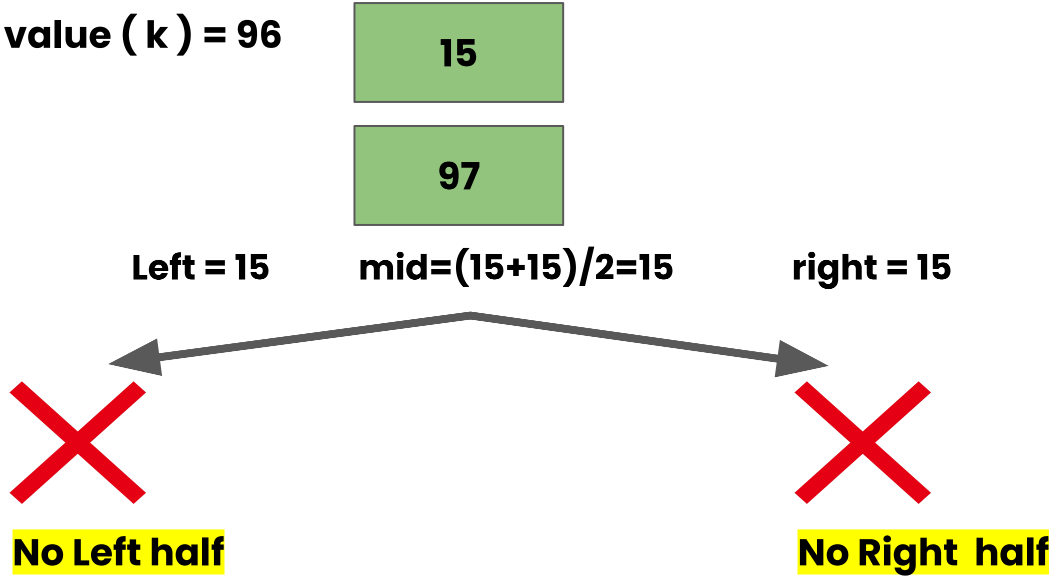
mid=(15+15)/2=15

right = 15

No Left half

No Right half

Let search key value (k) = 96



15

97

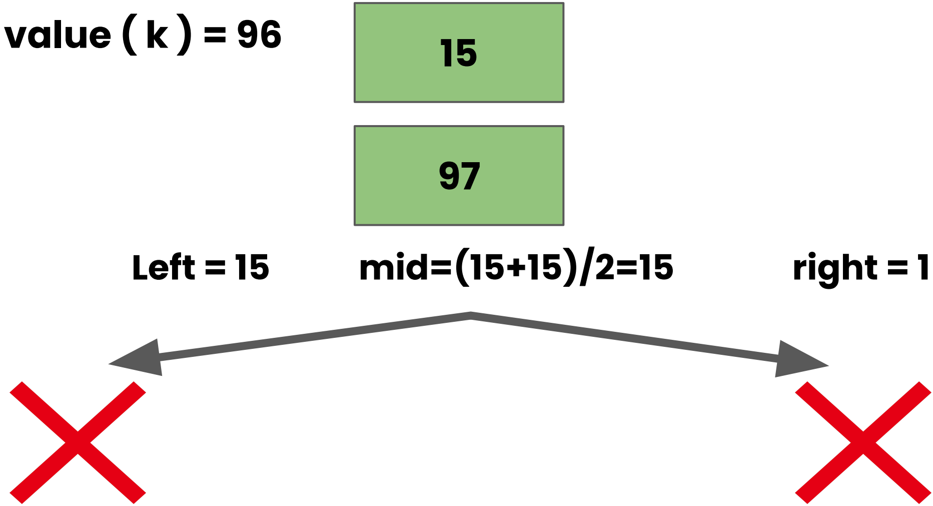
Left = 15 mid=(15+15)/2=15 right = 15

No Left half

No Right half

But Pointers will be updated → Left = 15 mid=(15+14)/2=14 right = 14

Let search key value (k) = 96



15

97

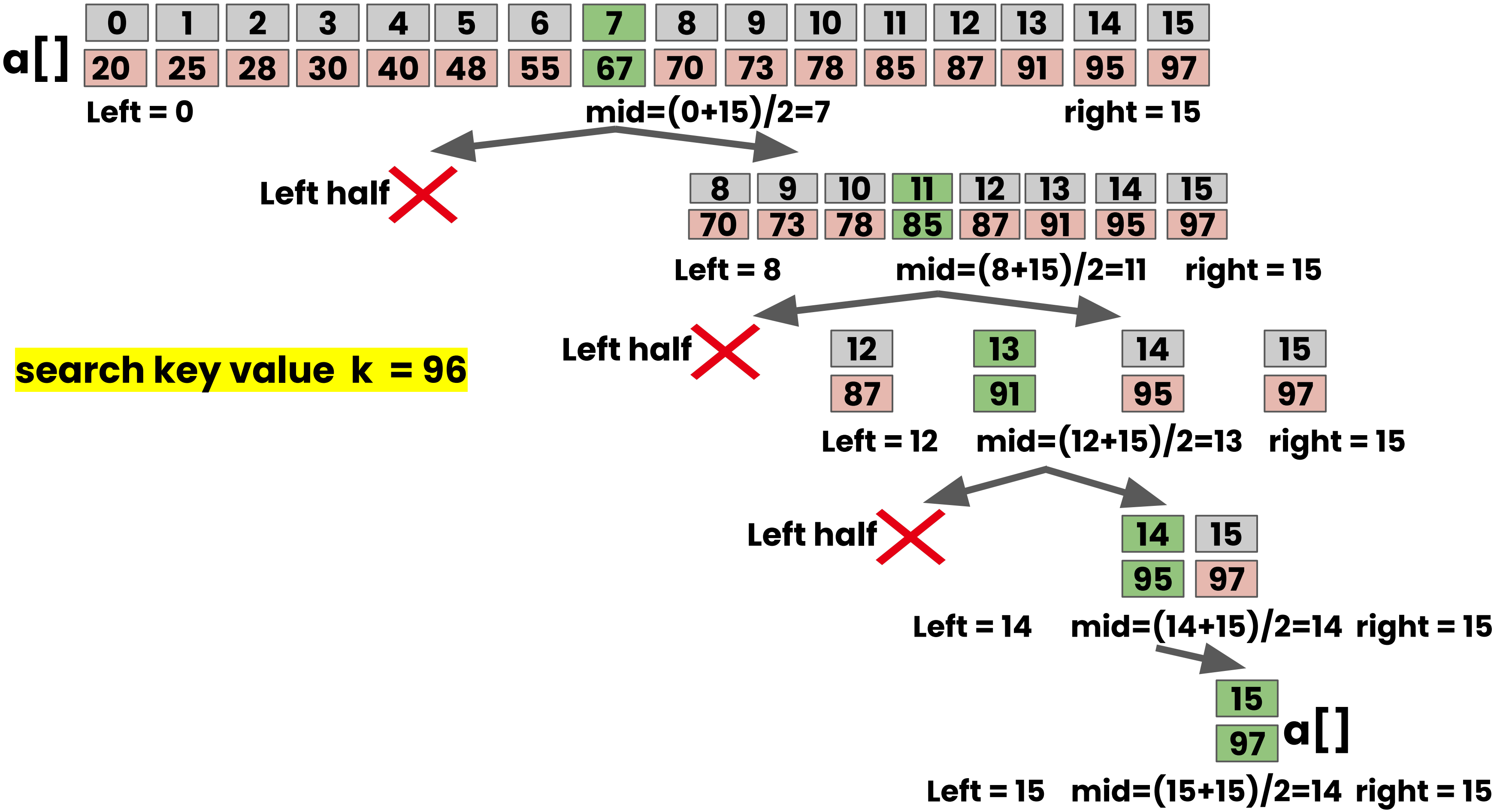
Left = 15 mid=(15+15)/2=15 right = 15

No Left half

No Right half

But Pointers will be updated ⇨ Left = 15 mid=(15+14)/2=14 right = 14

Stop Condition : left > right



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0 mid=(0+15)/2=7 right = 15

Left half

8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

Left = 8 mid=(8+15)/2=11 right = 15

search key value k = 96

Left half

12	13	14	15
87	91	95	97

Left = 12 mid=(12+15)/2=13 right = 15

Left half

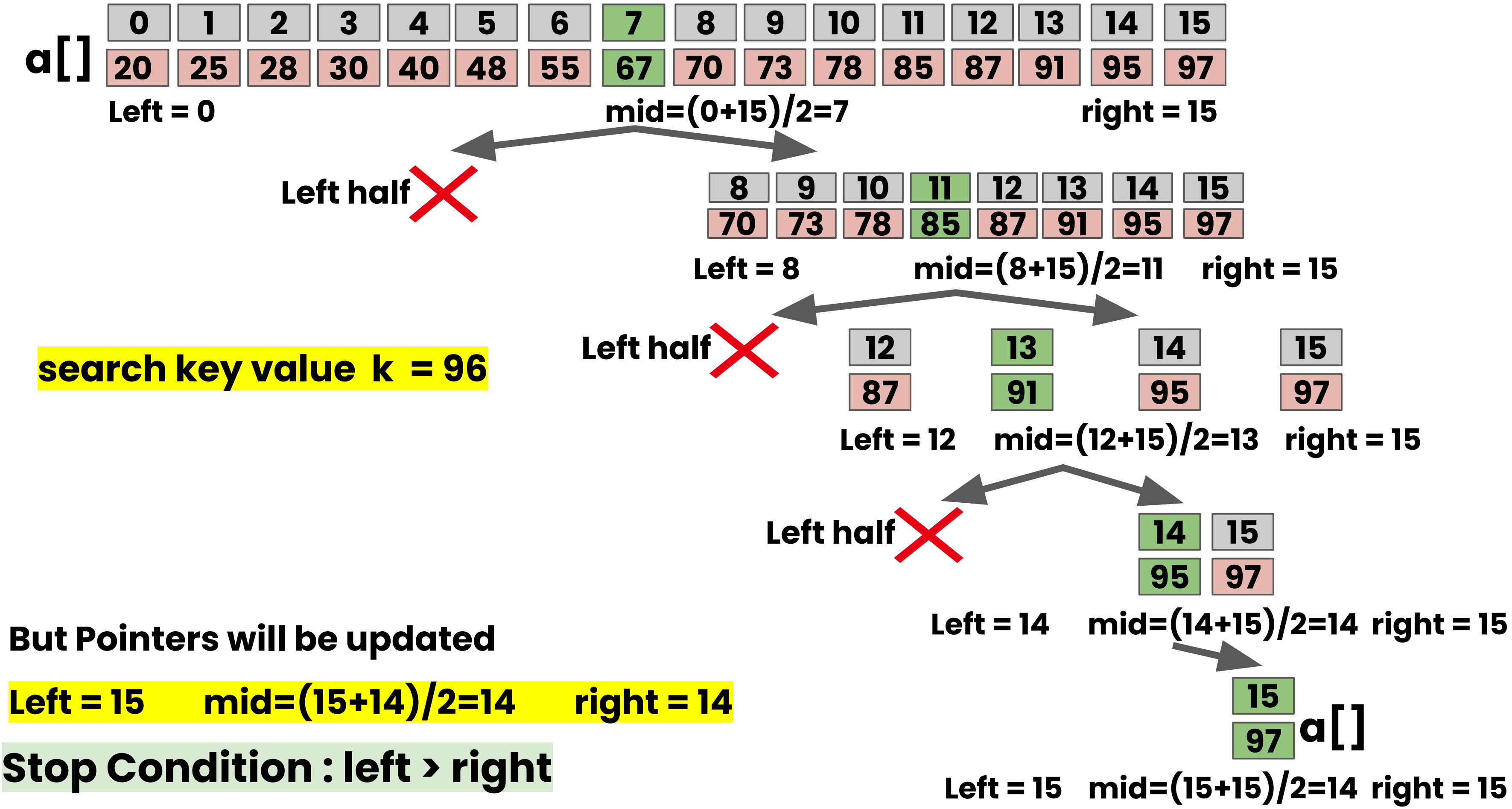
14	15
95	97

Left = 14 mid=(14+15)/2=14 right = 15

15
97

a[]

Left = 15 mid=(15+15)/2=14 right = 15



a[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0 mid=(0+15)/2=7 right = 15

Left half

8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

Left = 8 mid=(8+15)/2=11 right = 15

Left half

12	13	14	15
87	91	95	97

Left = 12 mid=(12+15)/2=13 right = 15

Left half

14	15
95	97

Left = 14 mid=(14+15)/2=14 right = 15

15
97

a[]

Left = 15 mid=(15+15)/2=14 right = 15

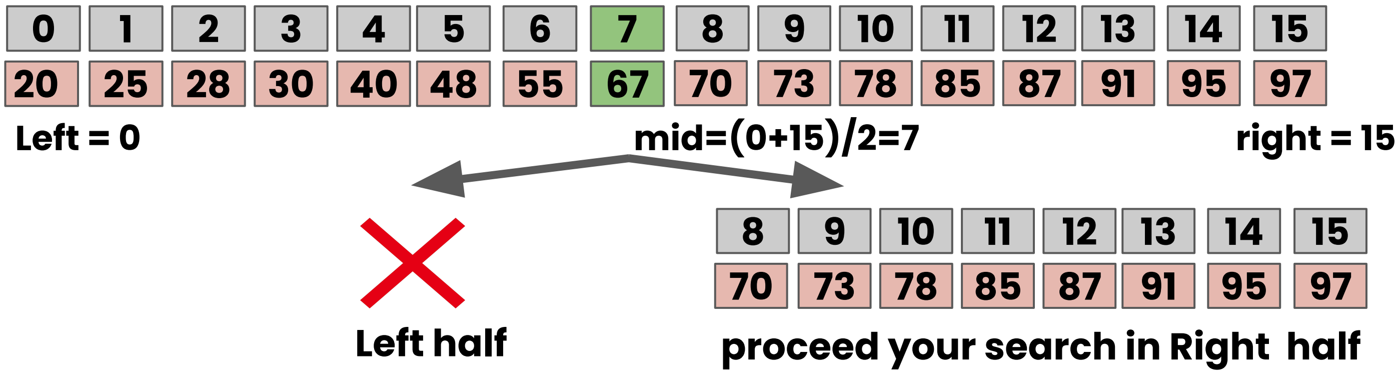
search key value k = 96

But Pointers will be updated

Left = 15 mid=(15+14)/2=14 right = 14

Stop Condition : left > right

Performance Analysis of Binary Search :



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	25	28	30	40	48	55	67	70	73	78	85	87	91	95	97

Left = 0

mid=(0+15)/2=7

right = 15

8	9	10	11	12	13	14	15
70	73	78	85	87	91	95	97

Left half

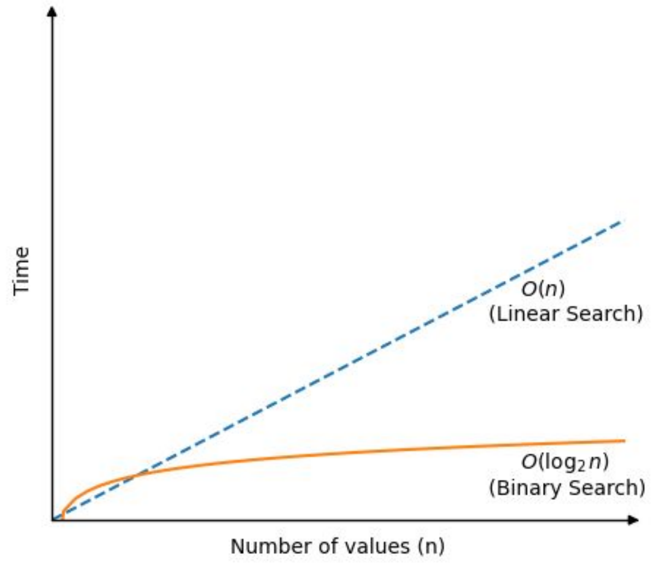
proceed your search in Right half

Recurrence Relation

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

$$T(n) = \mathbf{O(\log n)}$$

Complexity Analysis :

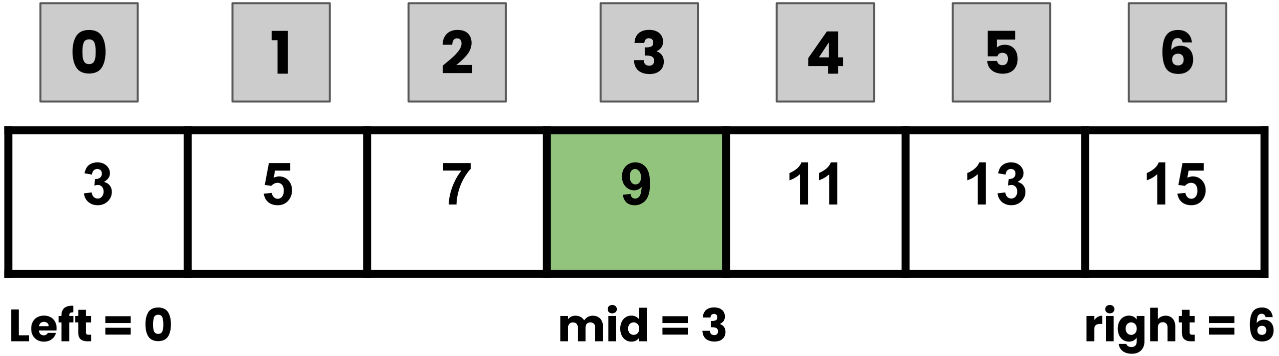


No. of comparisons = number of times the array size n can be halved.

Mathematically expressed as logn.

Visualizer Demo : Binary Search

Finding the middle index : (odd length)

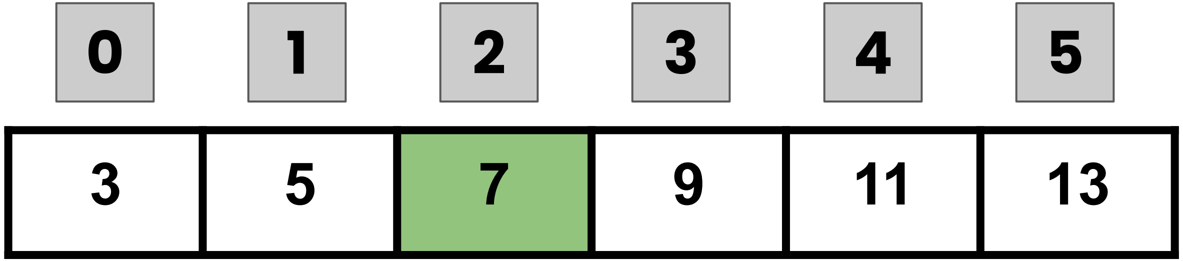


0	1	2	3	4	5	6
3	5	7	🟩 9	11	13	15

Left = 0 mid = 3 right = 6

Mid = (0 + 6) //2 = 3

Finding the middle index : (even length)



0	1	2	3	4	5
3	5	🟩 7	9	11	13

Left = 0 mid = 2 right = 5

Mid = (0 + 5) //2 = 2

Terminating conditions :

1. Target element is found
2. No element left to search (left > right)

References

- 6.2. Searching
- 6.3. The Sequential Search
  - 6.3.1. Analysis of Sequential Search
- 6.4. The Binary Search
  - 6.4.1. Analysis of Binary Search