Searching Algorithms

Linear Search

Binary Search

Linear Search

Linear Search

Idea of the algorithm is to iterate across the array from left to right, searching for a specified element.

In pseudocode:

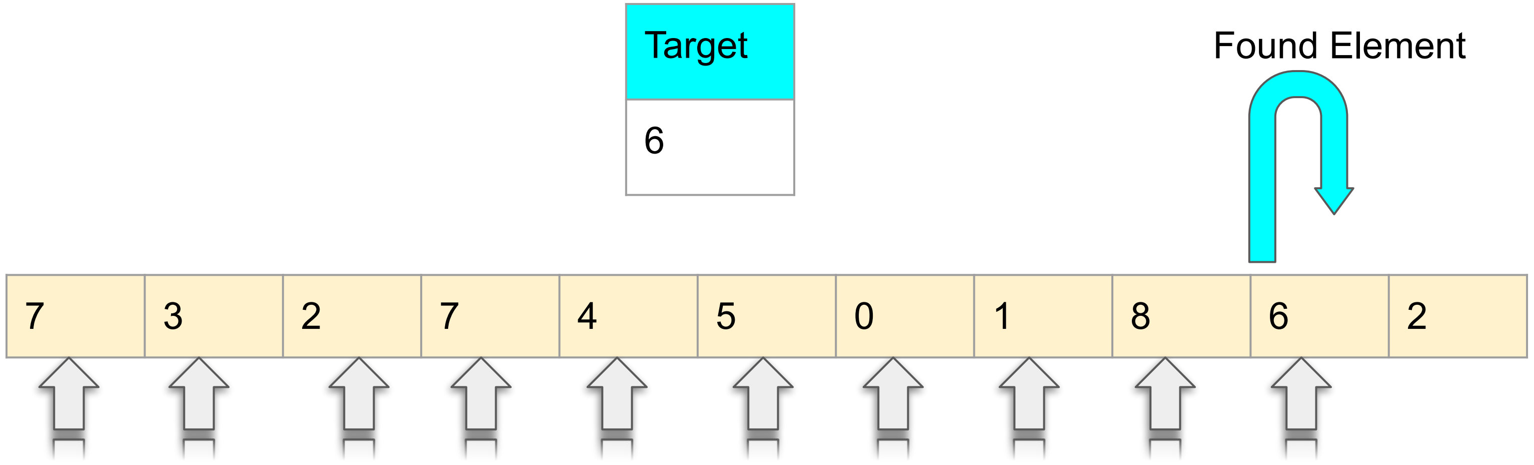
1. Repeat, starting at the first element:
    - ➢ If the first element is what you’re looking for (the target), stop.
    - ➢ Otherwise, move to the next element.
2. For i from 0 to n-1

```
        If i’th element is the element to be searched

                Return true

    Return false
```

Linear Search Demo



Running time of Linear Search

1. Worst Case: If the element to be searched is the last element in the array or if the element is not found in the array, we have to walk through the entire array n.

   O(n)

2. Best Case: If the searched element is the first element in an array, we don’t need to traverse through the array.

   O(1)

We need more efficient algorithm apart from Linear Search(O(n))

Binary Search

Binary Search

The idea of algorithm is to divide and conquer, reducing the search by half each time while trying to find the target element.

Important: To perform binary search in an array, the array needs to be sorted always.

Pseudocode:

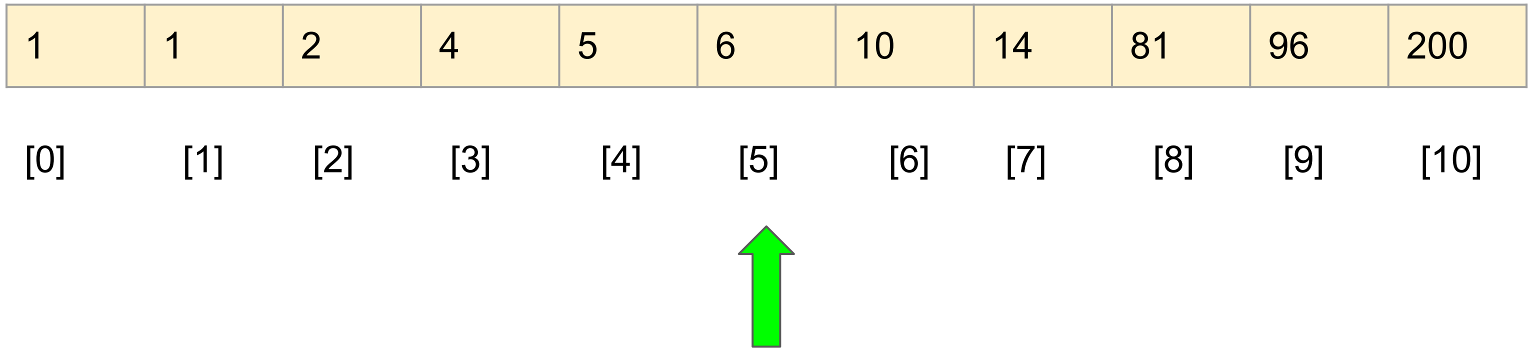
```
1.  If no items
        Return False
    If middle item is target item
        Return True
    Else If target item < middle item
        Search left half
    Else if target item > middle item
        Search right half
```

2. Repeat until the array is of size 0

   - Calculate the middle point of the current array.
   - If the target is at the middle, stop.
   - Otherwise, if the target is less than what’s at the middle, repeat, changing the end point to just to the left of the middle.
   - Otherwise, if the target is greater than what’s at the middle, repeat, changing the start point to just to the right of the middle.

Binary Search Demo

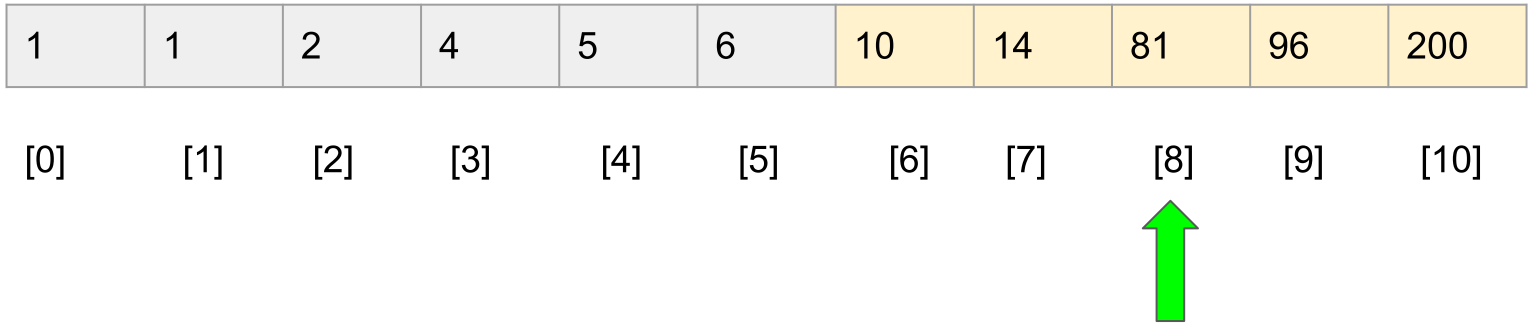
Target	Start	End	Middle
14	0	10	10/2 = 5



1	1	2	4	5	6	10	14	81	96	200
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]

Binary Search Demo

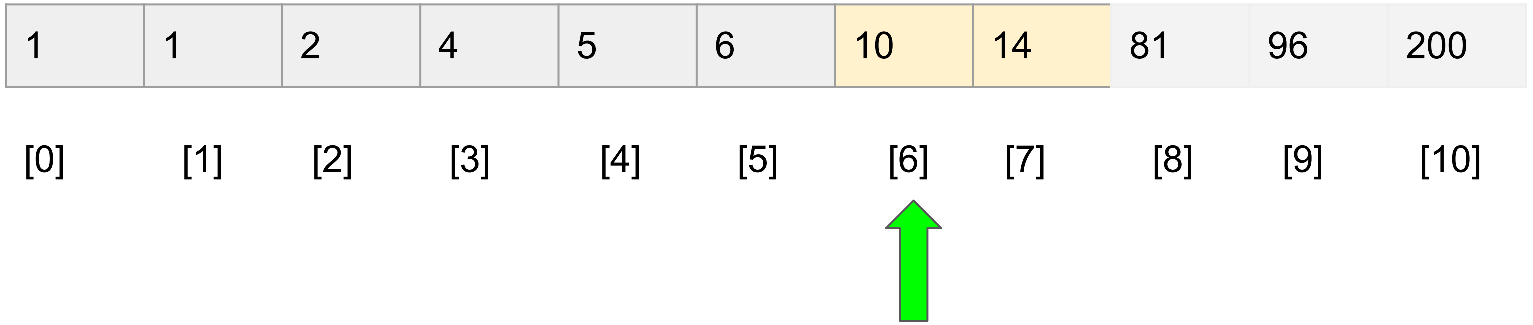
Target	Start	End	Middle
14	6	10	(6+10)/2 = 8



1	1	2	4	5	6	10	14	81	96	200
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]

Binary Search Demo

Target	Start	End	Middle
14	6	7	(6+7)/2 = 6



Binary Search Demo

Target	Start	End	Middle
14	7	7	(7+7)/2 = 7

1	1	2	4	5	6	10	14	81	96	200
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]

Found the target element

Running time of Binary Search

1. Worst-case: divide the list element repeatedly in halfs until we find the element or either the element is absent in the list.

   O(log n)

2. Best-case: When the target element is the midpoint of the given full array

   O(1)