sadā śiva samāramabhāṃ śaṅkarācārya madhyamām..
asmadācārya paryantāṃ vande guru paramparām..

*Salutation to the lineage starting with lord **Sadasiva**, with **Adi Sankara** in the middle and continuing up to my immediate teacher.*

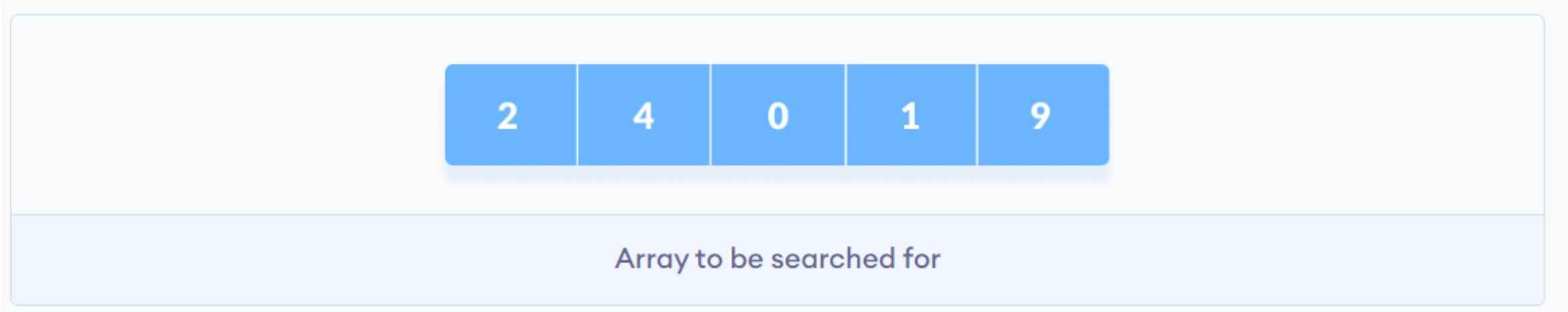
సదాశివుడు మొదలుకొని మధ్యలో ఆదిశంకరునితో మొదలై నా తక్షణ గురువు వరకు కొనసాగే వంశానికి వందనం.

DATA STRUCTURES ALGORITHMS

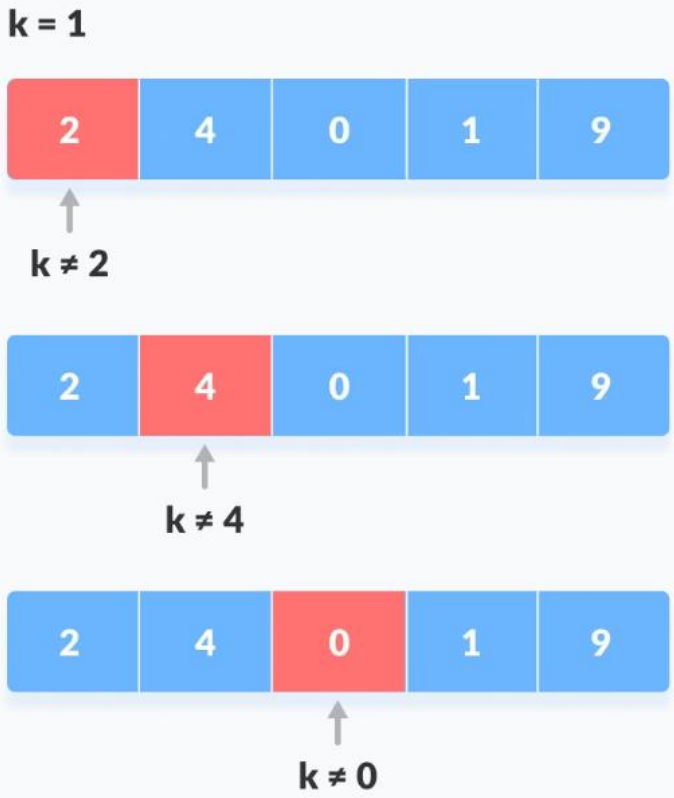
- Linear Search
- Binary Search
- Membership Operators
- Jump Search
- Fibonacci Search
- Exponential Search
- Interpolation Search

How Linear Search Works?

The following steps are followed to search for an element **k = 1** in the list below.

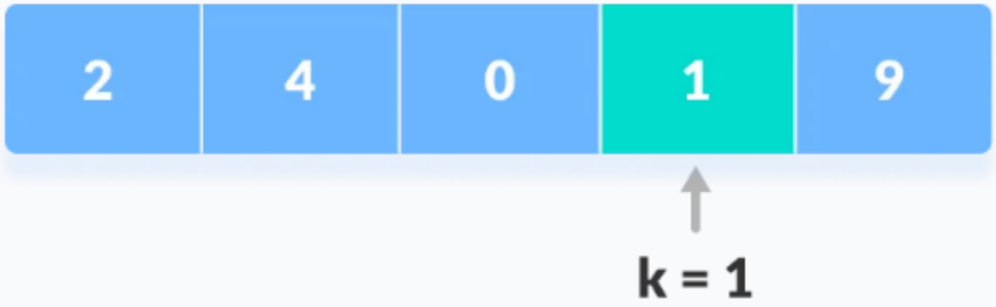


1. Start from the first element, compare `k` with each element `x`.



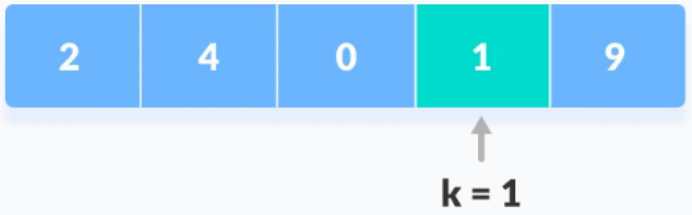
Compare with each element

2. If `x == k`, return the index.



Element found

2. If `x == k`, return the index.



Element found

3. Else, return `not found`.

Linear Search Algorithm

```
LinearSearch(array, key)
  for each item in the array
    if item == value
      return its index
```

Binary Search

Binary Search is a searching algorithm for finding an element's position in a sorted array.

In this approach, the element is always searched in the middle of a portion of an array.

> Binary search can be implemented only on a sorted list of items. If the elements are not sorted already, we need to sort them first.

Binary Search Working

Binary Search Algorithm can be implemented in two ways which are discussed below.

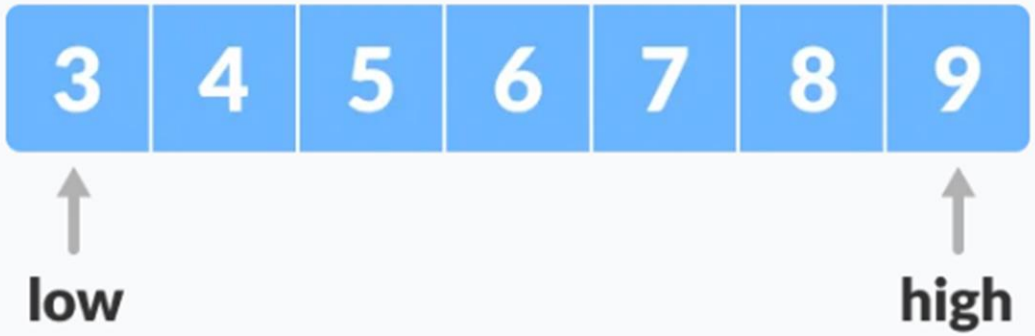
1. Iterative Method
2. Recursive Method

The recursive method follows the divide and conquer approach.

1. The array in which searching is to be performed is:

Let x = 4 be the element to be searched.

2. Set two pointers low and high at the lowest and the highest positions respectively.

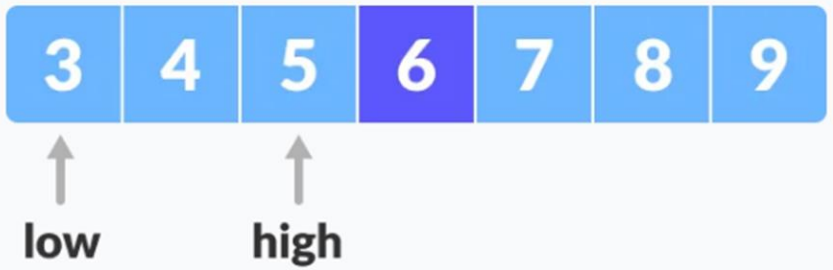


Setting pointers

3. Find the middle element `mid` of the array ie. `arr[(low + high)/2] = 6`.

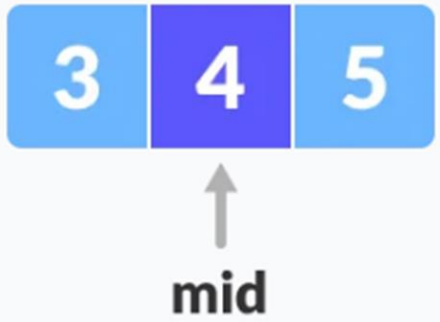
Mid element

4. If x == mid, then return mid.Else, compare the element to be searched with m.
5. If `x > mid`, compare `x` with the middle element of the elements on the right side of `mid`. This is done by setting `low` to `low = mid + 1`.
6. Else, compare `x` with the middle element of the elements on the left side of `mid`. This is done by setting `high` to `high = mid - 1`.



Finding mid element

7. Repeat steps 3 to 6 until low meets high.



Mid element

8. `x = 4` is found.

Found

Binary Search Algorithm

Iteration Method

```
do until the pointers low and high meet each other.
    mid = (low + high)/2
    if (x == arr[mid])
        return mid
    else if (x > arr[mid]) // x is on the right side
        low = mid + 1
    else                        // x is on the left side
        high = mid - 1
```

Binary Search Algorithm

Recursive Method

```
binarySearch(arr, x, low, high)
    if low > high
        return False
    else
        mid = (low + high) / 2
        if x == arr[mid]
            return mid
        else if x > arr[mid]         // x is on the right side
            return binarySearch(arr, x, mid + 1, high)
        else                                 // x is on the left side
            return binarySearch(arr, x, low, mid - 1)
```

Leet code Searching Problems

https://leetcode.com/tag/binary-search/