**Searching Algorithms**

Here, we’re going to discuss how to,

* Implement **linear search** on arrays
* Implement **binary search** on sorted arrays
* Implement a **naive string searching** algorithm
* Implement the **KMP string searching** algorithm

There are many different search methods on arrays in JavaScript:

indexOf

includes

find

findIndex

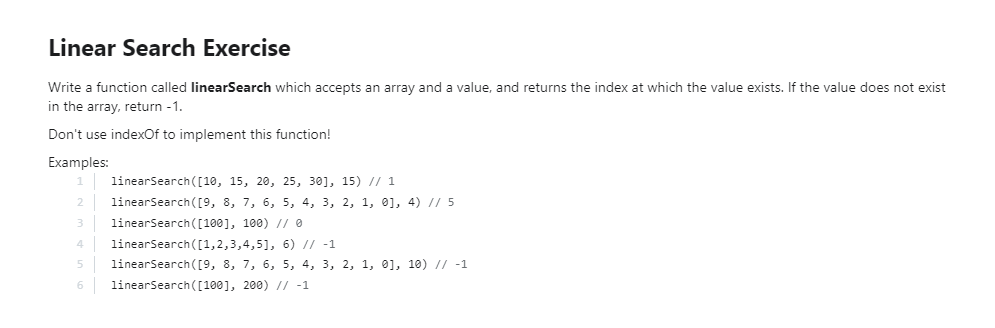
Linear Search Algo. Is working behind the scene of all these methods.

**Linear Search Algorithm:-**

**Pseudo Code:**

* So, write a function, call it whatever you want but accepts an array and a value
* Loop through the array and check if the current array element is equal to the value
* If it is, return the index at which the element is found
* If the value is never found, return -1

**Examples:**



function linearSearch(arr, val){

for (let i = 0; i < arr.length; i++) {

if(val===arr[i])  
 return i;

}

return -1;

}

linearSearch([9, 8,7,6,5,4,3,2,1,0], 4);

**BigO:**

Best Case: O(1)

Worst Case: O(n)

Average Case: O(n)

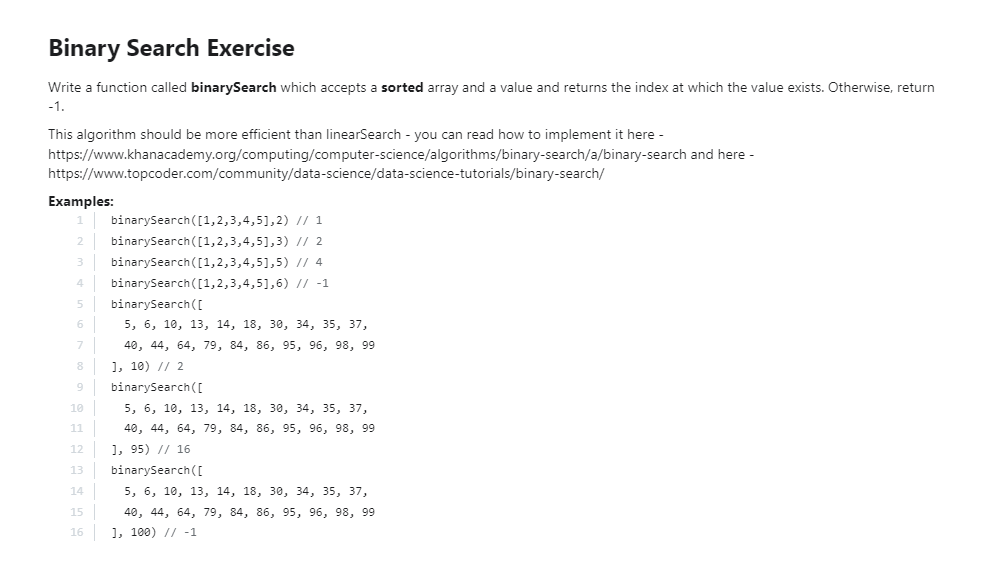
**Binary Search Algorithm:-**

* Binary search is a much faster form of search
* Rather than eliminating one element at a time, you can eliminate half of the remaining elements at a time
* Binary search **only works on sorted arrays!**
* It **follows Divide & Conquer** Design Pattern of Algorithm.

**Pseudo Code:**

* This function accepts a sorted array and a value
* Create a left pointer at the start of the array, and a right pointer at the end of the array
* While the left pointer comes before the right pointer:
* Create a pointer in the middle
* If you find the value you want, return the index
* If the value is too small, move the left pointer up
* If the value is too large, move the right pointer down
* If you never find the value, return -1

**Example:**



function binarySearch(arr, val){

let i=0,

j=arr.length-1;

while(i<j){

let mid = Math.round((i+j)/2);

if(arr[mid]>val){

j=mid+1;

}else if(arr[mid]<val){

i=mid-1;

}else{

return mid;

}

}

return -1;

}

binarySearch([5, 6, 10, 13, 14, 18, 30, 34, 35, 37, 40, 44, 64, 79, 84, 86, 95, 96, 98, 99], 95);

**BigO:**

Best Case: O(1)

Worst Case: O(logn)

Average Case: O(logn)

**Naïve String Search Algorithm:-**

**Pseudo Code:**

* Define a function that takes two strings (Longer & Shorter String).
* Loop over the longer string
* Loop over the shorter string
* If the characters don't match, break out of the inner loop
* If the characters do match, keep going
* If you complete the inner loop and find a match, increment the count of matches
* Return the count

**Example:**

function naiveSearch(lngStr, shtStr) {

let count=0;

for (let i = 0; i < lngStr.length; i++) {

for (let j = 0; j < shtStr.length; j++) {

if(shtStr[j]!==lngStr[i+j]){

break;

}

if(j===shtStr.length-1){

count++;

}

}

}

return count;

}

naiveSearch('lorie loled', 'lorz');