Explain the concept of Big O notation and analyse the time complexity of the linear and binary search algorithms.

What is Big O?

1. Big O notation is used to describe the efficiency of the algorithm.

2. This method provide the upper bound of the complexity by this we can find the worst case

of time and space complexity of the given algorithm.

\* CONSTANT - O(1)

\* LOGARITHMIC - O(log n)

\* LINEAR - O(n)

\* LINEARITHMIC - O(n log-n)

\* QUADRATIC - O(n^2)

\* CUBIC - O(n^3)

\* EXPONENTIAL - O(2^n)

\* FACTORIAL - O(n!)

Time complexity of linear search algorithm:

const ary = [55,87,36,91,2,64,32,48,101,529,6];

const find = 64;

function linearSearch(array){

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

        if(array[i] == find){

            return `${find} is present at index-${i} of the array [${array}]`

        }

    }

    return `${find} cannot be found in the array [${array}]`

}

console.log(linearSearch(ary));

In the above code we run the for loop until we find the search element.

In worst case it can go upto length of the array Which is O(n).

Time complexity of binary search algorithm:

const ary = [11,13,17,23,29,33,58,61,73,86,94];

const find = 13;

function binarySearch(array){

    let start = 0;

    let end = array.length-1;

    while(end > 0){

        let mid =  parseInt((start+end)/2);

        if(array[mid] == find){

            return  `${find} found at index-${mid} of the array [${array}]`

        }

        if(find < array[mid]){

            end = mid-1;

        }

        if(find > array[mid]){

            start = mid+1;

        }

    }

    return `${find} cannot be found in the [${array}]`

}

console.log(binarySearch(ary));

* In binary search we are using two pointers (start ,end) to set the boundary for searching the element. Where mid is the middle element found b/w the boundaries.
* On every iteration we check the middle element is equal to the search element or not. If true we return the index otherwise we alter the pointers and continue the process until we find the search element.
* With help of this method we don’t have to search the entire array to find the particular element instead we only search a part of the given array.
* So the time complexity is O(log n)