Algorithms: Binary Search Part-2

Relevel
by Unacademy

Table Of Contents

Searching Algorithms

In computer science, the algorithms which are used to find or search an element(s) in a given set of data are defined as searching algorithms. Searching algorithms are applied depending on the nature of the dataset. Here in this module, we would be focusing on two prime searching algorithms which are

- Linear search
- Binary search
- Ternary Search

Ternary Search

Like linear search and binary search, ternary search is a searching technique used to determine the position of a specific value in an array. In binary search, the sorted array is divided into two parts, while in ternary search, it is divided into three parts, and then you determine in which part the element exists. Ternary search, like binary search, is a divide-and-conquer algorithm. It is mandatory for the array (in which you will search for an element) to be "monotonically arranged" before you begin the search. In this search, after each iteration, it neglects ⅓ part of the array and repeats the same operations on the remaining ⅔.

Ternary Search

Let us consider the following example to understand the code.
Let the sorted array be ar[] = = {2,3,5,7,8,9,10,13,14} with indices from 0 to 8. You are required to find the position of x=13. In this array. Divide the sorted array into the following 3 parts by evaluating the values of mid1 and mid2.

1. {2,3,5}
2. {7,8,9}
3. {10,13,14}

Ternary Search

Here ar[mid1]=5 and ar[mid2]=10. As 13 is not equal to ar[mid1]and ar[mid2] and it is also not smaller than ar[mid1], you can safely assume that it lies in the 3rd part of the array as it is greater than ar[mid2].

Run the ternary search again with l=7 and r=8. Now, ar[mid1]=ar[7]=13 and ar[mid2]=ar[8]=14

As ar[mid1]=x, mid1 is the required answer. If the value is not in the array, it returns −1 as the answer.

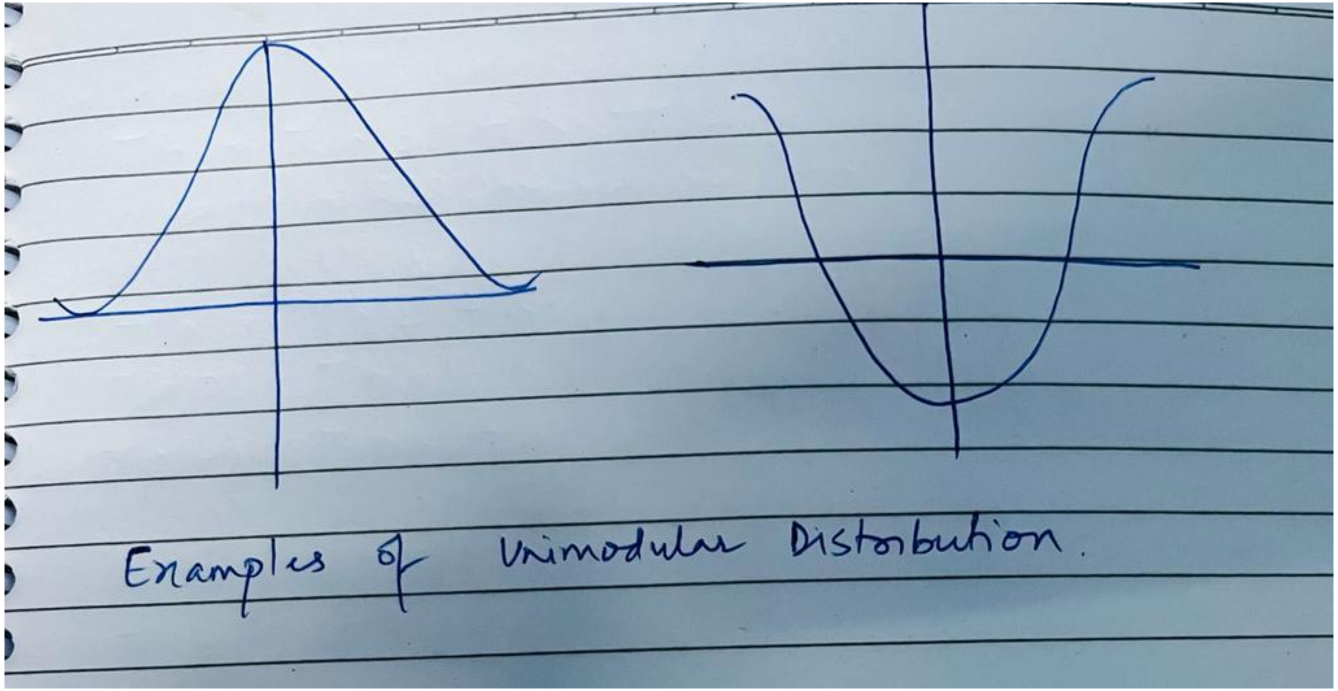
Code Link : https://ideone.com/ONHbLw

Ternary Search

Application of Ternary Search

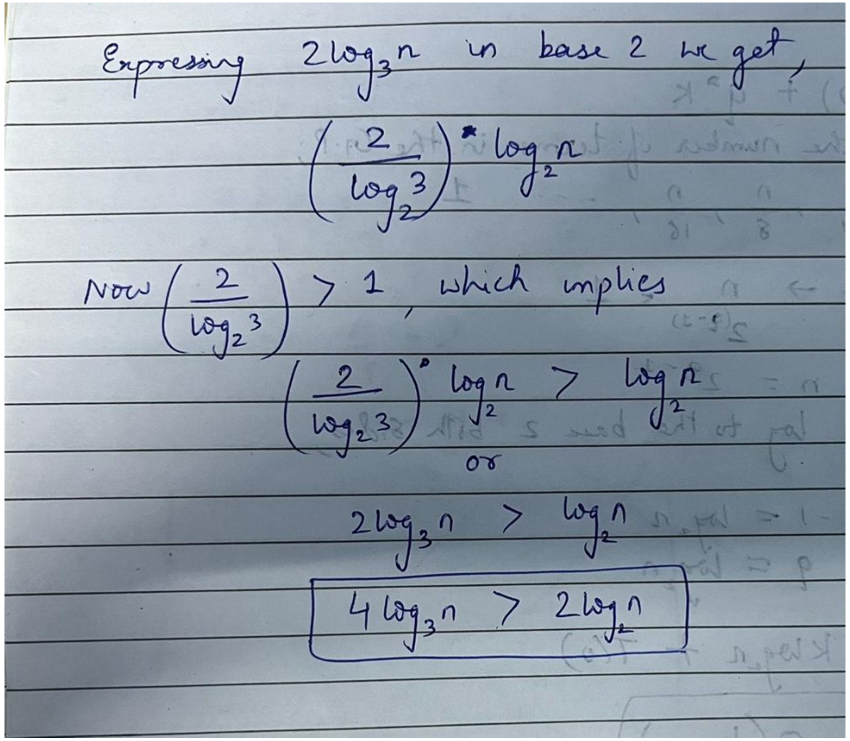
Talking about the utility of ternary search, this searching technique determines the maximum or minimum value of unimodal functions. Unimodal functions are functions that have a single highest value. The value increases at first reach a single peak, and then decreases. Following are examples of unimodal distributions.

Ternary Search



Why is Binary Search best !?

Hence proved that Ternary
Search is computationally more expensive than
Binary Search



Expressing $2\log_3 n$ in base 2 we get,

$$\left(\frac{2}{\log_2 3}\right) * \log_2 n$$

Now $\left(\frac{2}{\log_2 3}\right) > 1$, which implies

$$\left(\frac{2}{\log_2 3}\right) \log_2 n > \log_2 n$$

or

$$2\log_3 n > \log_2 n$$

$$\boxed{4\log_3 n > 2\log_2 n}$$

Binary Search Problems

Problem statement 1:

You are Given a series of positive integer numbers in the form of a sorted array as Ai where i is the index. N is the positive number which you have to check in the given array Ai whether the given N is duplicate in the Array or not. If it is duplicate output as “FALSE” if it is non duplicate output as “TRUE”.

Note: Use Binary search algorithm to find the N.

```
let arr = [1, 3, 5, 5, 7, 8, 9];
findNumber = 5;
```

Binary Search Problems

Javascript code: https://www.ideone.com/sSnpBH

```
let binarySearch = function (arr, findNumber, start, end) {
  if (start > end) return;

  let mid = Math.floor((start + end) / 2);

  if (arr[mid] === findNumber) {
    if (arr[mid - 1] === findNumber || arr[mid + 1] === findNumber) {
      return "TRUE";
    }
    return "FALSE";
  }
```

```
   if (arr[mid] > findNumber) {
     return binarySearch(arr, findNumber, start, mid - 1);
   } else {
     return binarySearch(arr, findNumber, mid + 1, end);
   }
};

let arr = [1, 3, 5, 5, 7, 8, 9];
let findNumber = 5;

console.log(binarySearch(arr, findNumber, 0, arr.length - 1));
```

Binary Search Problems

Problem statement 2:

Given an array of positive integers, and a value sum, determine if there is a subset of the given set with sum equal to given sum. Find the two numbers that summation will be equal to the value in the sum variable.

Input: arr = [1, 3, 4, 5, 7, 10, 11, 12] , sum = 9

Output: 5 4

There is a subset (4, 5) with sum 9.

Binary Search Problems

Javascript code: https://www.ideone.com/Z6NmMJ

```
let binarySearch = function (arr, findNumber, start, end) {
    if (start > end) return;

    let mid = Math.floor((start + end) / 2);
    if ((arr[mid] + arr[mid -1] === findNumber) || (arr[mid] + arr[mid +1]
=== findNumber)) {
        if ((arr[mid] + arr[mid -1] === findNumber)) {
            return [arr[mid], arr[mid-1]];
        }
        else {
            return [arr[mid], arr[mid+1]];
        }
    }
    if ((arr[mid] + arr[mid -1] > findNumber) || (arr[mid] + arr[mid +1] >
findNumber)) {
      return binarySearch(arr, findNumber, start, mid - 1);
    } else {
      return binarySearch(arr, findNumber, mid + 1, end);
    }
  };

  let arr = [1, 3, 4, 5, 7, 10, 11, 12];
  let findNumber = 9;

  console.log(binarySearch(arr, findNumber, 0, arr.length - 1).join(' '));
```

Practice Problem

1. Find square root of a number upto given precision using the binary search algorithm
2. Find the number of rotations in Rotated Sorted array

Thank You!