**DAY 4 ASSIGNMENT**

**Question 1**

**In the Binary Search algorithm, it is suggested to calculate the mid as**

**beg + (end - beg) / 2 instead of (beg + end) / 2. Why is it so?**

ANS:

Binary search(at least the way I implement it) relies on a simple property - a predicate holds true for one end of the interval and does not hold true for the other end

int beg = 0; // pred(beg) should hold true

int end = n;// length of an array or a value that is guranteed to be out of the interval that we are interested in

while (end - beg > 1) {

int mid = (end + beg) / 2;

if (pred(a[mid])) {

beg = mid;

} else {

end = mid;

}

}

// answer is at a[beg]

**Question 2**

**Write the algorithm/function for Ternary Search.**

Ans:

Ternary search is a divide-and-conquer search algorithm. It is mandatory for the array (in which you will search for an element) to be sorted before we 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.

Algorithm

The steps involved in this algorithm are:

(The list must be in sorted order)

Step 1: Divide the search space (initially, the list) in three parts (with two mid-points: mid1 and mid2)

Step 2: The target element is compared with the edge elements that is elements at location mid1, mid2 and the end of the search space. If element matches, go to step 3 else predict in which section the target element lies. The search space is reduced to 1/3rd. If the element is not in the list, go to step 4 or to step 1.

Step 3: Element found. Return index and exit.

Step 4: Element not found. Exit.

Complexity

Worst case time complexity: O(log N)

Average case time complexity: O(log N)

Best case time complexity: O(1)

Space complexity: O(1)