Problem 1: - ([DPV] 2.16)

Here we are given an infinite array A[.], where n entries are different sorted integers. We can call each element on array by index i on array A. Rest elements are filled with infinity. We must find element x in array. First, we will find lower and upper bound by searching on array by increasing index with 2 power i followed by binary search to solve this problem.

Step1: -

We will start with base case where index of array starts with index i=1.

Then we will find the upper bound for x and check if x equals, greater or less than x , by increasing the index of Array A[.] by $A = 2^i \; such \; that$, A[2],A[4],A[8],.. ..until x is found or we reached infinity. From this, we will get the range for lower(l) and upper bound(u) of array A where x can be in between that.

Step2: -

We will check if A(u) is equals to x , otherwise

From the lower and upper bound from step 1 , we will apply binary search starting from lower bound(l) until upper bound (u) to find the element x.

Step3:-

If element found from step 2.

Return true

Otherwise

Return false.

Correctness: -

As array is sorted and we are increasing upper bound ranged by 2^i and then we are either finding x or comparing if its lies within upper bound range. So from step 1, once we have found the upper bound range where x can lie in array we did binary search to find element x on this range.

Runtime:-

Step 1 for finding the upper bound range will for n take max O(log n) time for n size integers . Then on Step2 Binary search take O(log n) time. So total run time of above algorithm will be O(log n):-

O(log o)+O(log n) = O(log n)

Problem2:-

Here we will used modified binary search to find the element A(i)=2i+5 where 1<=i<=n .We will used modified binary search to find such a element on array A of index i.

Initialized lower bound to 0 . Upper bound for binary search is n . Key to found is 2i+5.

Step1.)

Perform binary search starting from index 1 until n and find element A(i). Here we have modify the search criteria to find key such that if A(i)=2i+5 . Find middle value mid from the array using (high -low)/2 at index i . Check if we found the key on array A[mid]= 2i+5.

Step2.)

If such element not found from step 1 , then divide the array into left and right array L and R from index midpoint mid.

Step3)

Now check condition if A[mid] >2i+5, then repeat step1,2 on right array R otherwise repeat steps 1,2 on left array L.

Step4) If we found such element A[i]=2i+5 then return yes otherwise return no.

Correctness: -

Here as we have used customized binary search where our key to be found is 2i+5. So during this, we are checking if A[mid] is equal to our question key 2i+5

Runtime: -

As we have used binary search here to find the required key so total run time from step 1 to 3 is O(log n) and step 4 time is O(1). So total run time of above algorithm will be O (log n)