Xiaoka Xiang

Homework 2.

Due: Monday, September 13, 2021 before 8am EDT.

Problem 1 ([DPV] 2.16)

This problem is similar to Binary search, which is O(log(n)).

To make it a binary search, we need to first identify an array of known size. The array we are looking for will be an array which has lower bound less than x, and upper bound is infinite or greater than x. within this range, we perform binary search. The search process should be in a way with complexity of no more than O(log(n)).

The search process is then:

Compare A[0] with x, if A[0] == x, then result is 0, if A[0] > x, result is null.

Otherwise, we then compare A[1] with x, if A[1] == x, then result is 1, if A[1] > x, result is null.

Otherwise, we compare A[2] with x, if A[1] == 2, then result is 2, if A[2] > x, result is null.

Otherwise, starting from A[2], the next value we will compare against is not A[i+1], but A[2^i], so the next value we will compare against will be A[4] here. Once we find an A[i] such that, A[i]< x< A[i^2], or A[i^2] = infinite, the we have a known array which upper bound is A[i^2], and lower bound is A[i].

For .i= 2, once A[2] is less than x, we are comparing A[4], if A[4] >x, we have A[2]<x<A[4] and we can use A[2] and A[4] as the lower and upper bound of our array for binary search, if not we will then check A[8].

At each step, if A[2^i] = x we return the index.

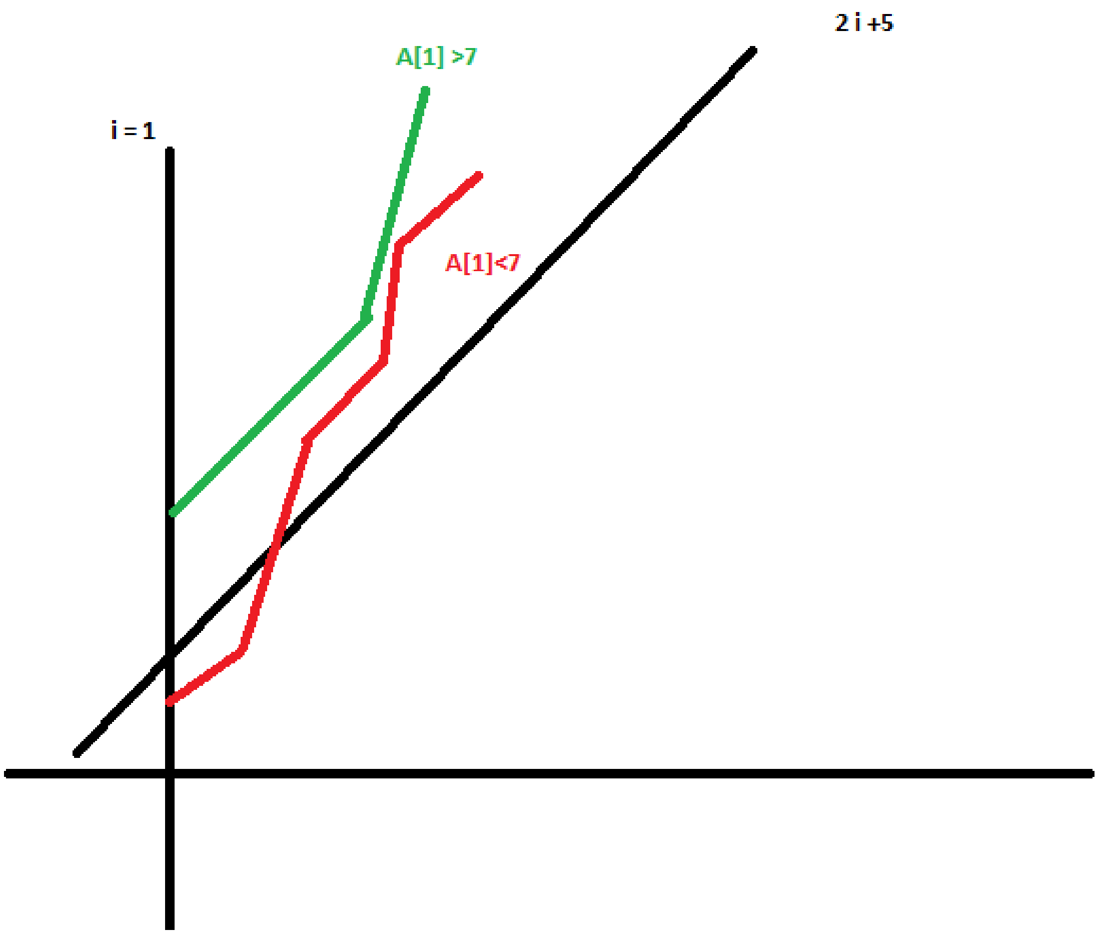
The search process is of complexity less than O(log(n)), and binary search is also O(log(n)) at most, so the entire process is O(log(n)).

Problem 2 (Index function)

The infinite array is different number, sorted, and odd. So for any give A[i] and A[i+1], below formula holds:

A[i+1] > = A[i]+2.

If we draw a line with all points, the slope between any points will be greater or equal to 2.



As a result, we have:

1. If the first element A[1] is greater than 7, there will be no element fulfill this A[i] = 2i+5, result is always “no”, complexity is O(1).
2. If first element A[1] is less than or equal to 7,
3. Let .i = n/2 first, then compare 2i + 5 vs A[i], if A[i] < 2i +5, then 2i + 5 is in range of A[i] to A [n]. If A[i] > 2i+ 5, then 2i+ 5 is in range of A[1] to A[i].
   Next step is to take the range with the lower bound’s value is smaller than 2i+5, upper bound’s value is greater than 2i+5, compare the middle point of such array vs 2i+5, and only keep the half that has 2i+5 between the new upper and lower point. The mid point’s index is .i = n/4, n/4^2,….

Eventually, since it is all odd number, our range becomes A[a] to A[a+1]. We will check Either our A[a] or A[a+1] be our 2a+5, if so then the return is "Yes", if A[a]< 2a+5< A[a+1], or A[n] < 2n+5 then return "No".
Say we have n = 32, and A[1] = 5, A[2] = 9, A[3] = 13, we first compare A[16] vs 37, we will get A[16] > 37 > A[1]. Then we compare A[8] vs 21, and find A[8]> 21 > A [1], then A[4] vs 13, we will find A[4] >13> A[1], then A[2] vs 9, we will find that A[2] = 9, and return yes.

Since each time we do i/2, each time we compare only half of the array we will have, the complexity is then O(log(n))