L36
Binary Search : Problem Solving 2

Join Discord - https://bit.ly/ly-discord

RECAP

More Binary Search Practice

Below is the code for binary search, right?

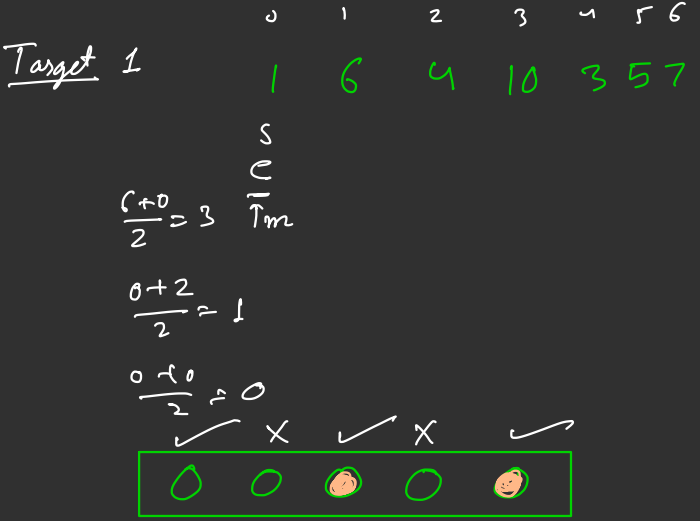
```
boolean findTarget(int arr[], int n, int target) {
    int beg = 0, end = n - 1;
    while(beg <= end) {
        int mid = (beg + end)/2;
        if(arr[mid] == target)  return true;
        else if(arr[mid] > target) end = mid - 1;
        else beg = mid + 1;
    }
    return false;
}
```

Will run for an array sorted in non-decreasing order

What if the array is not sorted?

Binary Searchable Elements

The problem is that given an array containing unique integers, you need to find the number of array elements which can be searched using the binary search algorithm shown in the previous slide.



Example:

Input
[1, 6, 4, 10, 3, 5, 7]

Output
3

Explanation:
1, 6 and 10 will be searchable, others won't be.

Let’s think.
Hint : Try to think simple, really simple.

Implementation

Time Complexity?

Part 2

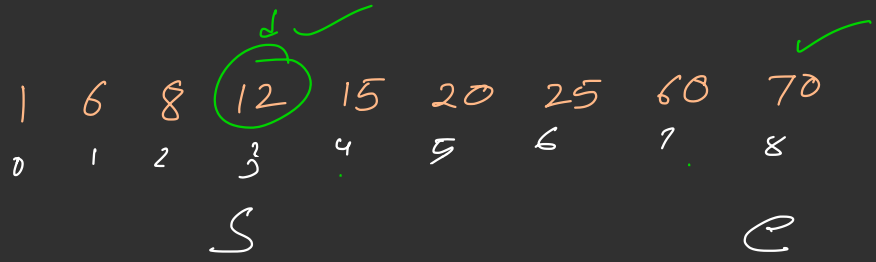
What if we use a modified binary search to find an element in a sorted array?

```
boolean findTarget(int arr[], int n, int target) {
    int beg = 0, end = n - 1;
    while(beg <= end) {
        int mid = random_between(beg, end);
        if(arr[mid] == target)  return true;
        else if(arr[mid] > target) end = mid - 1;
        else beg = mid + 1;
    }
    return false;
}
```

Will run for an array sorted in non-decreasing order

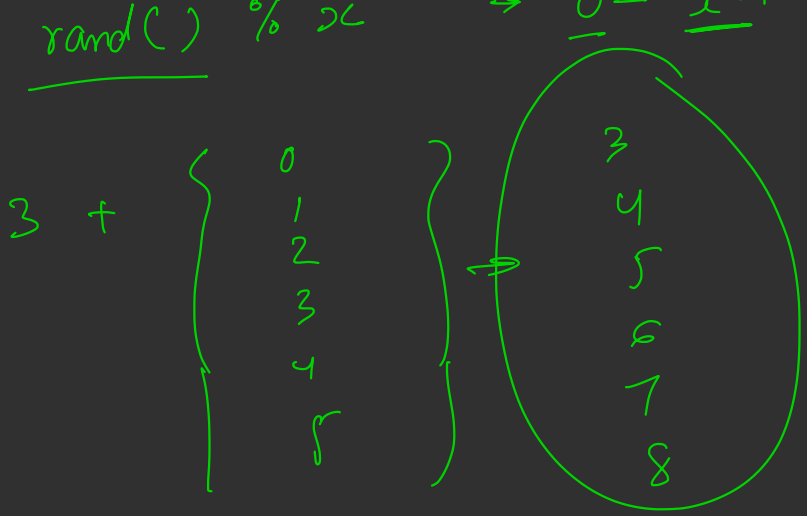
If the array is unsorted?

May or may not work.



3 — 6

rand()

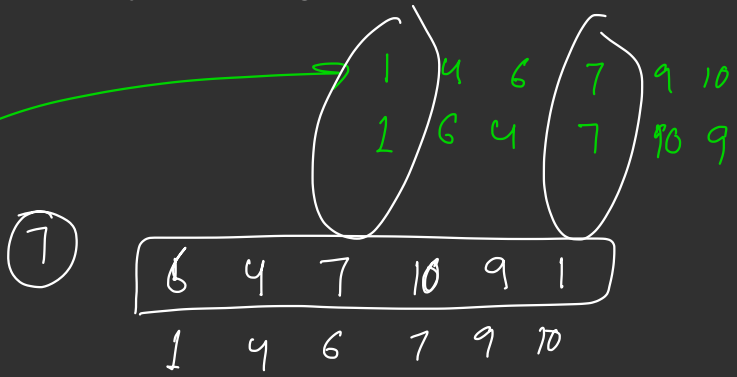


Modified Binary Searchable Elements

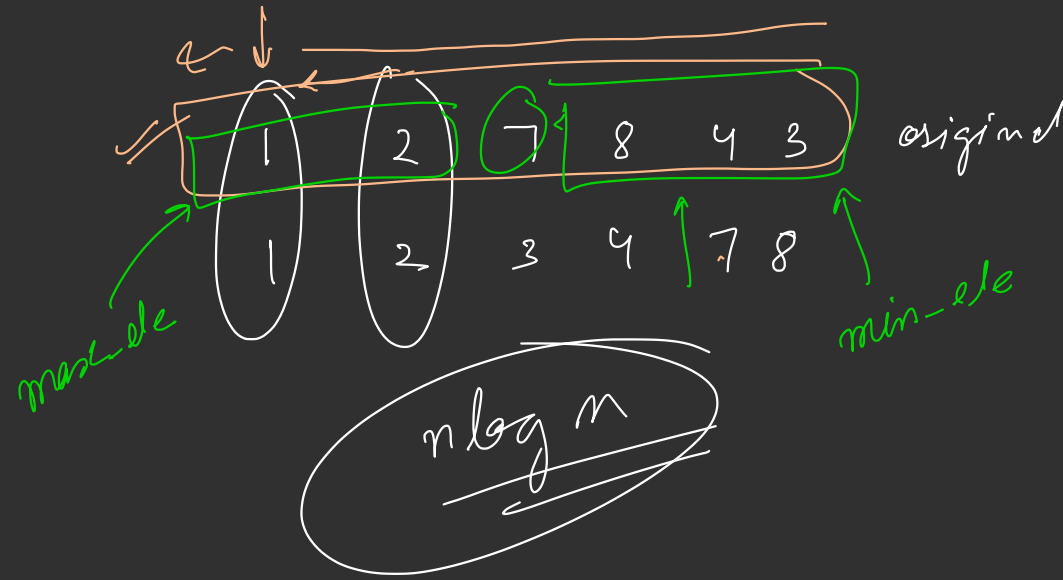
The problem is that given an array of unique integers, we need to find how many array elements will **always** be searchable using the modified binary search algorithm?

Example:

Input
[1, 6, 4, 7, 10, 9]
s e
Output
2

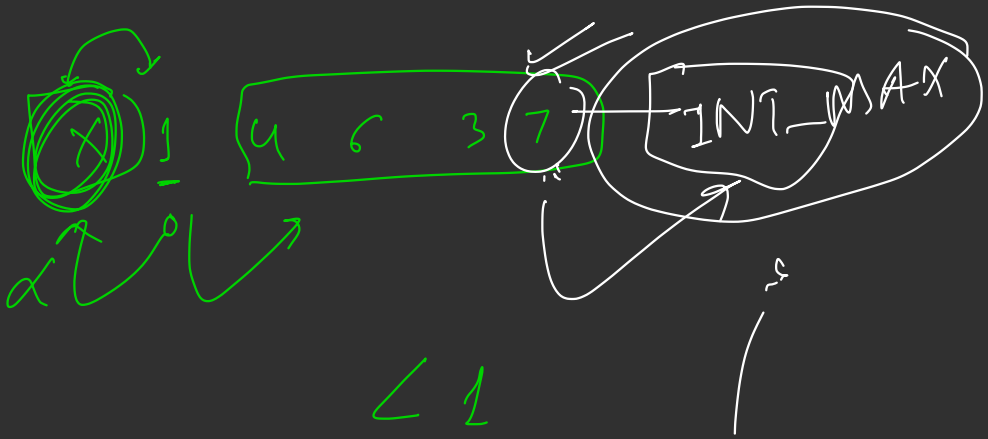


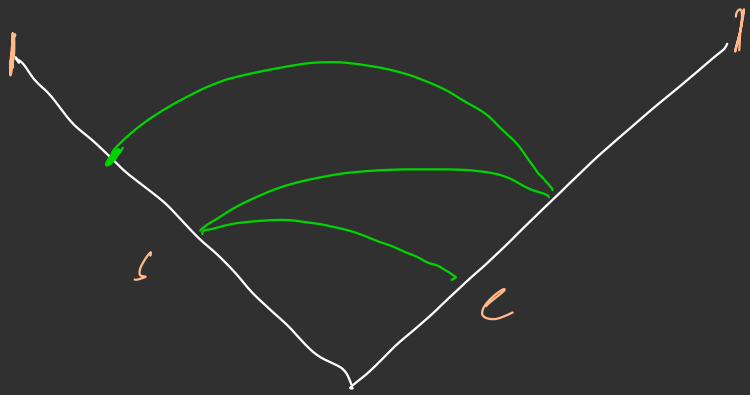
Explanation:
Only the elements 1 & 7 will **always** be searchable. All others may or may not be searchable depending on the random mid values chosen.



9 6 4 7 10 1

1 4 6 7 9 10



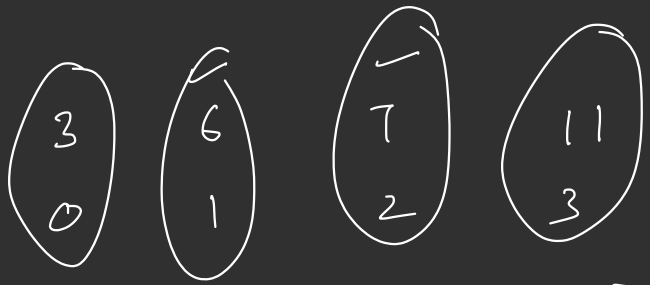


Let’s think

Implementation

Next problem:
Koko Eating Bananas

Given a value of k, can we find the minimum hours required to finish all bananas?



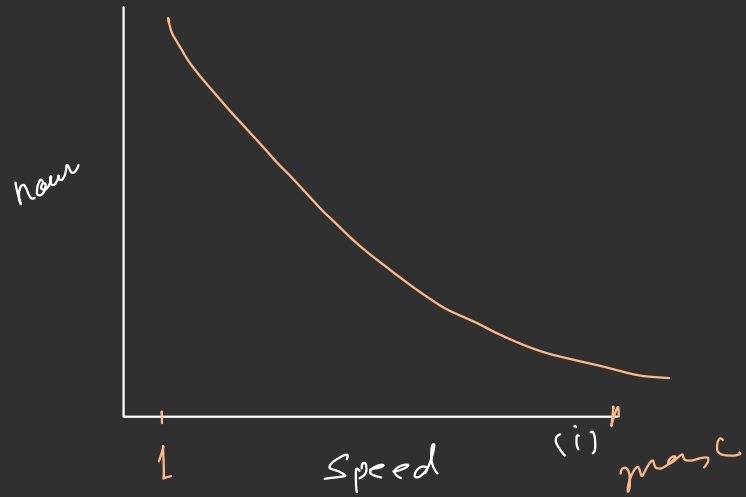
Speed	hours
1	27
2	15
3	10
4	

1 + 2 + 3 + 4

10

11 - 3
8 - 3
5 - 3
2 - 2
0

9



Is there monotonicity anywhere?

Solution

Time Complexity

Thank You!

Reminder: Going to the gym & observing the trainer work out can help you know the right technique, but you'll muscle up only if you lift some weights yourself.

So, PRACTICE, PRACTICE, PRACTICE!