UMC201: Binary Search And Formalisation

Sirjan Hansda

8th August 2023

Question. **We Optimise the Binary Search Code as Follows:**

INPUT:Array A, len(A)=n, target=x

```
p = 0, q = n − 1
while(p<q):
    m = ⌊(p+q)/2⌋
    if x≤A[m]:
        q = m
    else:
        p = m + 1
if(x == A[p])
    return p
else:
    return(−1)
```

OPTPUT: -1 or position of x

(a) Why does this code not work for this modification?

```
if x<A[m]
    q=m
else:
    p=m+1
```

(b)Find the fix for the code and Prove that the fix works:

```
    m = ⌈(p+q)/2⌉
    if (x ≤ A[m])
        q = m
    else:
        p = m + 1
```

Answer. (a) The Code does not work for the given modification because, while making changes to p and q we are not checking for A[m] itself. Even if A[m]=x, then, we directly change p to m+1, thereby skipping over A[m]

Example: A=[1,3,5,10,15] and x=5. so, at first, p=0 and q=4, therfore m=2. Now, A[m]=x, but because of the modified logic, p is changed to m+1=3 .
Thus now, p=3 and q=4. Obviously, 5 is not present between them. Therfore, the given code fails to find the target value, even though it is in the array.

Answer. (b) The code is not accurate.
To Prove this, we provide a counter-example:
A=[1,3,7,11,25,34] and target=3.
So, in the first iteration: p=0, q=5. So, m=3. A[3]=11 $\geq$ 3, so we change q=3. For the second iteration, p=0 and q=3.
Now, for second iteration m=2. Again, A[2] = 7 $\geq$ 3. So, we change, q=2. Thus for 3^{rd} iteration, p=0 and q=2.
Again, in this iteration, m=1. So, A[m]=3, but, by the given logic, we again set q=1. For the fourth iteration, p=0 and q=1.
In this iteration again we find m=1, and the same effect continues. Infact, this loop will never exit, thereby being unable to give the correct answer.

Proposed Fix:

p=0,q=n−1
...
$m = \lceil \frac{p+q}{2} \rceil$ #Unchanged
if $(x < A[m])$:
    q=m−1
else:
    p=m

Proof that the new Algorithm works:

LOOP INVARIANT:
(I)$A[0:p] \leq x$ OR p=0
(II) $x < A[q+1:n-1]$

INITIATION: Initially, p=0 and q=n-1. So, 1^{st} invariant is satisfied. A[n:n-1] is empty array, and hence the 2^{nd} invariant statements is vacuously true.

MAINTAINANCE: We assume that the loop invariants are satisfied. Therefore, we enter the loop:

(i) $0 \leq p < q \leq n-1$ ...from the loop condition and the initial values of p,q

$$\text{(ii) } A[0:p-1] \leq x \quad \ldots \text{from } 1^{st} \text{ invariant}$$
$$\text{(iii) } A[q+1:n-1] > x \quad \ldots \text{from } 2^{nd} \text{ invariant}$$

Now, we compute $m = \lceil \frac{p+q}{2} \rceil$. From basic mathematics, we know that, $p < m \leq q$. Now, we divide into 2 cases:

1) $x < A[m]$: Then

$$q' = m - 1$$
$$p' = p$$

Since, $x < A[m]$ and because A is a sorted array, so, $x < A[m : n-1] \Rightarrow x < A[q'+1 : n-1]$. Thus, the second loop invariant holds true. The first invariant also holds true, as the value of p remains same as the previous value. Thus, the invariants hold true.

2) $x \geq A[m]$: Then

$$p' = m$$
$$q' = q$$

Since, $x \geq A[m]$ and because A is a sorted array, so, $x \geq A[0:m] \Rightarrow x \geq A[0:p']$.Therefore, the 1^{st} invariant holds true. The second loop invariance also holds true, as no modification has been done to q.

TERMINATION: When the loop terminates, $p = q$ and furthermore, going by the loop invariant, $A[0:p] \leq x < A[p+1:n-1]$. Thus, if $x \in A$, then $x = A[p]$. Atlast, the algorithm checks if $A[p] = x$. If yes then we have found x otherwise, we have failed to do so. Thus, the invariance is maintained even after the loop exits. **Thus the algorithm is correct**.

Proof that the given algorithm terminates:
The loop terminates only when p=q or p>q.

What happens after every step is that, either $p' = \lceil \frac{p+q}{2} \rceil$ or $q' = \lceil \frac{p+q}{2} \rceil - 1$. Thus,
Initial Length=$q - p + 1$
Final Length= $(\lceil \frac{p+q}{2} \rceil - 1) - p + 1 = \lceil \frac{p+q}{2} \rceil - p \ldots$ if q is modified.
Final Length= $q - (\lceil \frac{p+q}{2} \rceil) + 1 = q - \lceil \frac{p+q}{2} \rceil + 1 \ldots$ if p is modified.
Now, we know,

$$p < \lceil \frac{p+q}{2} \rceil \leq p$$

So, no matter what happens, the length of the array always decreases. And we know, that only integers are acceptable as indices of arrays. **And since the list of integers have a least element, we are sure that at sometime, the length of the array must come down to 0.** Thus the loop must terminate.