Problem

Input: A sorted array A with size n and $value$.
Output: An index i such that $A[i] = value$. If such an index does not exist, the result is implementation-defined.

Algorithm

```
lower_bound(A[0 : n), value)
    first = 0
    count = n
    while count > 0
        count_half = count // 2
        mid = first + count_half
        if A[mid] < value
            first = mid + 1
            count = count - count_half - 1
        else
            count = count_half
    return first
```

Notation

For the following proof, we have some predefined notations.

- For any array A, any indices l and r, $A[l, r)$ represents the subarray of A containing the element $A[l], A[l+1], \ldots, A[r-2], A[r-1]$, excluding $A[r]$ (hence the open interval on r). For the case where $l \geq r$, $A[l, r)$ is defined as an empty array.
- For any array A, any indices l and r, and any number $value$, we write $A[l, r) < value$ to represents that every element in the subarray $A[l, r)$ is less than $value$.

  $$A[l, r) < value \quad \Longleftrightarrow \quad \forall i \in [l, r),\ A[i] < value.$$

  Other comparison operators ($>, \leq, \geq$, etc.) work the same as above.
  Note that $A[l, r) < value$ is a tautology when $A[l, r)$ is empty.

Correctness

Expected Input.

An array A of size n sorted in non-decreasing order. And a number $value$ to search.

Expected Output.

Return the first index i such that $value \leq A[i]$. That is,

$$A[0, i) < value \leq A[i, n).$$

If no such index exists (i.e., all elements are less than $value$), the procedure returns n:

$$A[0, n) < value.$$

Proof Assumption.

For the purpose of this correctness proof, we assume the existence of a conceptual sentinel element $A[n] = \infty$. This assumption does not affect the algorithm's execution because:

1. The algorithm only accesses indices in the range $[0, n)$ during execution
2. The loop invariant ensures $first + count \leq n$, so $mid < n$ always holds
3. The sentinel merely guarantees that an element $\geq value$ always exists, allowing us to unify the analysis of both cases (element found vs. not found)

With this sentinel, we can state our postcondition uniformly: the algorithm returns the smallest index $i \in [0, n]$ such that $value \leq A[i]$.

Intuition.

Let's first define a predicate P to clarify what we are dealing with.

$$P(i) : A[i] < value$$

That is, when we apply this predicate along all the indices on the sorted array A, we gets

$$T, T, \ldots, T, F, F, \ldots, F.$$

And we want to find the index of the first F (the first element that is NOT less than $value$). Note that F is guaranteed to exist because we set up a sentinel element $A[n] = \infty$. The algorithm works by maintaining a searching range, where the answer lies within, and repeatedly narrowing down this range until the range only contains one element, the exact answer.

Proof.

We prove by using the loop invariant $\mathcal{I}$. Let's define it as the following three properties.

- Search boundary check:
$$0 \leq first \leq first + count \leq n \tag{1}$$

- P is true for all elements before $first$:
$$\forall i \in [0, first), \quad P(i) = T. \tag{2}$$

- P is false for all elements from $first + count$ onward:
$$\forall i \in [first + count, n], \quad P(i) = F. \tag{3}$$

The loop invariant ensures that the index of the first F lies in the range $[first, first + count]$.

1. **Initialization:**

   At the start of the first iteration, we have $first = 0$ and $count = n$. The boundary check is satisfied. P is true before $first$ since there is simply no element before this. And P is false for the index $first + count = n$ since $A[n] = \infty$. Thus, $\mathcal{I}$ holds at the start.

2. **Maintenance:**

   Assume that $\mathcal{I}$ is true at the start of an iteration, we want to show that $\mathcal{I}$ still holds at the end of this iteration.

   Let's first find out where mid is. We have $count > 0$ by the termination condition, so

   $$mid = first + \lfloor count/2 \rfloor \quad \Rightarrow \quad first \leq mid < first + count.$$

   Now we consider the if-statement branches. To aviod confusion, we use a separate pair of notation for the variables we update: $first$ and $count$ represent the value of the variables at the start of the iteration, while $first'$ and $count'$ represent that at the end of the iteration.

   (a) If $P(mid) = T$, cut the searching range to $[mid + 1, first + count]$ as the following.
   At line 8, we have $first' = mid + 1$. Since we know that P holds for all element before $first$ by the assumption, and now we know that P also holds for all element in between $first$ and mid, then after this line, Property (2) still holds:

   $$\forall i \in [0, first'), \quad P(i) = T.$$

Then at line 9, we update $count'$ by ensuring that the new $first' + count'$ still points to the same place as the original $first + count$ do.

$$\begin{aligned} first' + count' &= (mid + 1) + (count - \lfloor count/2 \rfloor - 1) \\ &= (first + \lfloor count/2 \rfloor + 1) + (count - \lfloor count/2 \rfloor - 1) \\ &= first + count \end{aligned}$$

Thus, after this line, we still have Property (3) since this right bound does not change and is already established by the assumption.

For Property (1), we already find out the range of mid, so it is easy work.

$$0 \le first \le mid < first + count \le n$$

$$\Rightarrow 0 < mid + 1 \le first + count \le n$$

$$\Rightarrow 0 < first' \le first' + count' \le n$$

(b) If $P(mid) = F$, we cut the searching range to $[first, mid]$.

At line 11, we update $count' = \lfloor count/2 \rfloor$, so that the new right bound is $first + count' = mid$. With the assumption, we have that now P is false for all the element from $first + count'$ onward. Thus, Property (3) holds.

And since we does not touch $first$, Property (2) still holds.

For Property (1), trivial work can show that

$$0 \le first \le first + \lfloor count/2 \rfloor < first + count \le n$$

$$\Rightarrow 0 \le first \le first + count' < n$$

In both branches, the three properties are all satisfied. Therefore, $\mathcal{I}$ is true at the end of this iteration.

3. **Termination:**

First, let's show that the loop can correctly terminate. In each iteration, we see that $count$ gets cut down in half in both branches. More carefully speaking, with the fact that

$$\lfloor count/2 \rfloor \le (count - 1)/2,$$

and the update value $count' \in \{\lfloor count/2 \rfloor, \lfloor count/2 \rfloor - 1\}$, this gives

$$0 \le count' < count.$$

Thus, $count$ must hit 0 at some point, which means that the loop always terminates.

When the loop terminates, we have $count = 0$. And based on the initialization and maintenance of $\mathcal{I}$, we establish that P is true for all element before $first$, and false for all element from $first$ onward.

$$\forall i \in [0, first), \quad P(i) = T,$$

$$\forall i \in [first, n], \quad P(i) = F.$$

Therefore, $first$ is the correct answer, the index of the first F.