3. Given a *sorted* array of n distinct integers $A[1, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give an algorithm that runs in time $O(\log n)$ for this problem.

   Solution:

```
Input: A sorted array $A$
Result: $i$ such that $A[i] = i$, if such an $i$ exists
Let $k = 1, j = n$;
while $j - k > 1$ do
    Set $\ell = \lfloor \frac{j+k}{2} \rfloor$;
    if $A[\ell] = \ell$ then
        Output $\ell$.
    else if $A[\ell] > \ell$ then Set $j = \ell$;
    ;
    else Set $k = \ell$;
    ;
end
if $A[k] = k$ then
    Output $k$;
else if $A[j] = j$ then Output $j$ ;
;
else Output "No such index";
;
```

Algorithm 2: Binary Search

Analysis: If $A[\ell] > \ell$, then it must be the case that any index i with $A[i] = i$ is in the interval $[k, \ell]$. Similarly, if $A[\ell] < \ell$, it must be the case that the index we want is in the interval $[\ell, j]$. Thus the above algorithm halves the size of the interval we are looking in, in each run of the while loop.

Runtime: The runtime satisfies: $T(n) \leq T(n/2) + O(1)$. Thus $T(n) \leq O(\log n)$.