Here are several problems that are easy to solve in $O(n)$ time, essentially by brute force. Your task is to design algorithms for these problems that are significantly faster.


1.A. Describe a fast algorithm that either computes an index $i$ such that $A[i] = i$ or correctly reports that no such index exists.

1.B. Suppose we know in advance that $A[1] > 0$. Describe an even faster algorithm that either computes an index $i$ such that $A[i] = i$ or correctly reports that no such index exists. (Hint: This is really easy.)

2. Suppose we are given an array $A[1..n]$ such that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that an element $A[x]$ is a local minimum if both $A[x-1] \geq A[x]$ and $A[x] \leq A[x+1]$. For example, there are exactly six local minima in the following array:

```
9 ▲ 7 7 2 ▲ 1 3 7 5 ▲ 4 7 ▲ 3 3 ▲ 4 8 ▲ 6 ▲ 9
```

Describe and analyze a fast algorithm that returns the index of one local minimum. For example, given the array above, your algorithm could return the integer 9, because $A[9]$ is a local minimum. (Hint: With the given boundary conditions, any array must contain at least one local minimum. Why?)

3. Suppose you are given two sorted arrays $A[1..n]$ and $B[1..n]$ containing distinct integers. Describe a fast algorithm to find the median (meaning the $n$th smallest element) of the union $A \cup B$. For example, given the input

$A[1..8] = [0, 1, 6, 9, 12, 13, 18, 20]$  $B[1..8] = [2, 4, 5, 8, 17, 19, 21, 23]$

your algorithm should return the integer 9. (Hint: What can you learn by comparing one element of $A$ with one element of $B$?)

To think about later:

4. Now suppose you are given two sorted arrays $A[1..m]$ and $B[1..n]$ and an integer $k$. Describe a fast algorithm to find the $k$th smallest element in the union $A \cup B$. For example, given the input

$A[1..8] = [0, 1, 6, 9, 12, 13, 18, 20]$  $B[1..5] = [2, 5, 7, 17, 19]$  $k = 6$

your algorithm should return the integer 7.