Assignment 08

Alex Clemmer
Student number: u0458675

The array indices are going up by one starting at 0, and the values are going down. We want to find the intersection, if any. So the answer is a binary search, more or less. The comparator basically operates by comparing the index `i` to the value at `A[i]`.

This problem is made simply by the fact that `i` and `A[x]` are both in ascending order, and that `i` is sequential. This means that if `A[i]` is larger than `i`, then `i` will never catch up, ever. Thus, if `A[i]` ¿ `i`, we have gone too far in the array (assuming it's there), and if `A[i]` ¡ `i`, we have not gone far enough (again assuming it's there). In other words, this is the crucial feature that allows us to build the comparator.

This allows us to build a binary search. We split array. If the midpoint is `A[i]` ¿ `i`, we split the left part; if `A[i]` ¡ `i`, we split to the right. By repeatedly doing this as we did in the last assignment, we can find the place where `A[i]` = `i`. If the `low - high` ¡ 1, we did not find the element.

Why is this log n? Simple: each time we get rid of half the array. How many times does it take to partition an array of n elements? $\log_n A = B$ is $n^B = A$. So it takes roughly $\log_2 n$ tries. Hence (O log n). If we don't know the length of the array to begin with, the last assignment showed that finding the end is log n, and 2 log n is still $O(\log n)$.