CPSC 320: Intermediate Algorithm Design and Analysis
Assignment #7, due Thursday, March 29$^{\text{th}}$, 2012 at 11:00

[15] 1. Consider the problem of taking a **sorted** array A containing distinct integers, and determining whether or not there is a position i such that $A[i] = i$.

[6] a. Describe a divide-and-conquer algorithm to solve this problem. Your algorithm should return such a position if it exists, or `false` otherwise. If $A[i] = i$ for several different integers i, then you may return any one of them.

Solution: The algorithm described here will find if such a position exists between two positions `first` and `last` of the array, including the endpoints. The idea is simple: we look at the middle position, and then recurse on the either the first half or the second half of the array depending on the result of the comparison.

```
Algorithm findPosition(A, first, last)

if (first = last) then
    if (A[first] = first) then
        return first
    endif
    return false
endif

mid ← (first + last)/2
if (A[mid] = mid) then
    return mid
endif

if (A[mid] < mid) then
    return findPosition(A, mid+1, last)
else
    return findPosition(A, first, mid-1)
endif
```

[6] b. Prove the correctness of your algorithm. That is, show that it will always return a value of i for which $A[i] = i$, unless no such value exists in which case it will return `false`.

Solution: We prove the correctness of the algorithm by induction on n, where n is the number of elements of `A` from position `first` to position `last`. Clearly the algorithm will return the correct answer if $n = 1$, since this is the case where `first` = `last`.

So suppose that $\texttt{first} < \texttt{last}$. If $\texttt{A}[\texttt{mid}] = \texttt{mid}$ then the algorithm will return the correct position. Consider now the case $\texttt{A}[\texttt{mid}] < \texttt{mid}$.

Claim 1 *For every non-negative integer $j \leq$ mid, $A[mid - j] <$ mid - j.*

Proof: By induction on j. When $j = 0$, we are comparing $\texttt{A}[\texttt{mid}]$ to $\texttt{mid}$, which is true since this is the case we are examining. Suppose now that the claim holds for j. Because A contains elements that are distinct and sorted,

$$\texttt{A}[\texttt{mid} - (j+1)] \leq \texttt{A}[\texttt{mid} - j] - 1 < (\texttt{mid} - j) - 1 = \texttt{mid} - (j+1).$$

QED

Thus, a position i such that $\texttt{A}[i] = i$ can not satisfy $i \leq \texttt{mid}$, and hence the solution can only be found in between positions `mid+1` and `last`.

Finally, we need to consider the case where $\texttt{A}[\texttt{mid}] > \texttt{mid}$. The proof of this case is symmetric to the previous one, and completes the induction step and the proof of the theorem.

[3] c. Analyze the running time of your algorithm as a function of the number of elements of A.

Solution: The running time of the algorithm satisfies the recurrence relation

$$T(n) \leq \begin{cases} T(n/2) + \Theta(1) & \text{if } n \geq 2 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

We have:

- The number of sub-problems at a given level i is 1.
- The size of the sub-problem at a given level i is $n/2^i$, however, note that the amount of work required to solve the only one sub-problem at level i is constant $\Theta(1) = c$.
- Therefore the total amount of work at level i is constant $\Theta(1) = c$.
- The total number of levels required in the algorithm is $log_2(n)$.

so we get:

$$\begin{aligned} T(n) &\leq T(\frac{n}{2}) + c \\ &= \sum_{i=1}^{log_2 n} c \\ &= c \cdot log_2 n \end{aligned}$$

Our algorithm therefore requires $\mathcal{O}(log_2(n))$ time.