Analyzing the Running Time of a Simple Recursive Algorithm

A Sample Assignment

Consider the following computational problem, which was also considered in the assignment for Reading #2:

First Nonzero Entry in Part of an Array

Precondition: An integer array `A`, with positive length `n`, and integers `low` and `high`, such that $0 \leq$ `low` $\leq$ `high` $\leq$ `n` $- 1$, are given as input.

Postcondition: If at least one of

$$\texttt{A[low]}, \texttt{A[low+1]}, \dots, \texttt{A[high]}$$

is nonzero, then $\texttt{A}[i]$ is returned as output, where i is the smallest integer such that `low` $\leq i \leq$ `high` and $\texttt{A}[i] \neq 0$. The value 0 is returned otherwise.

Consider, as well, the following recursive algorithm, which was also considered in the assignment mentioned above:

```
integer firstNonZero ( integer[] A, integer low,
                                              integer high ) {
1. if (low == high) {
2.   return A[low]
   } else {
3.   integer mid := floor((low + high)/2)
4.   integer firstChoice := firstNonZero(A, low, mid)
5.   if (firstChoice != 0) {
6.     return firstChoice
     } else {
7.     return firstNonZero(A, mid+1, high)
     }
   }
```

If you completed that assignment then you proved that this algorithm correctly solves the above computational problem.

1. Write a ***recurrence*** for the maximum number $T_{\mathsf{firstNonZero}}(k)$ of steps used by the above recursive algorithm, as a function of $k = \mathtt{high} - \mathtt{low} + 1$, for $k \geq 1$ — using the uniform cost criterion when doing so.

2. Guess a ***solution*** for this recurrence, that is, an expression for $T_{\mathsf{firstNonZero}}(k)$ that is not in the form of a recurrence.

3. Prove that your guess is correct.