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 \( \text{low} \) and \( \text{high} \), such that \( 0 \leq \text{low} \leq \text{high} \leq n - 1 \), are given as input.

Postcondition: If at least one of

\[
A[\text{low}], A[\text{low} + 1], \ldots, A[\text{high}]
\]

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

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

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

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

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

3. Prove that your guess is correct.