CMPS 102 — Fall 2018 — Homework 2

*"I have read and agree to the collaboration policy." - **Kevin Wang***

Solution to Problem 2: Finding the Peak

Let $A[1:n]$ be an array of $\mathbb{Z}^+$, where $n \geq 3$. Array A has special properties: $A[1] \leq A[2]$ and $A[n-1] \geq A[n]$. We define $A[x]$ as a *peak* if $A[x] \geq A[x-1]$ and $A[x] \geq A[x+1]$.

Algorithm 1 Uses divide-and-conquer to locate a peak in an array

```
FIND-PEAK (A , n):
Let x = n/2
if A[x − 1] ≤ A[x] ≤ A[x + 1] then
    A[x] is a peak. Return index x.
else if A[x − 1] ≥ A[x] then
    FIND-PEAK (A[0 : x − 1] , x)
else
    {When A[x + 1] ≥ A[x]}
    FIND-PEAK (A[x : n] , x)
end if
```

Claim 1. *The time complexity of the algorithm is $O(\log n)$.*

Proof. It takes time $O(2)$ to check the neighbors of an array element. The time complexity of each reduced half array is $T\left(\frac{n}{2}\right)$. Therefore, the recurrence relation for the algorithm is defined as:

$$T(n) = T\left(\frac{n}{2}\right) + O(2)$$

Thus, by Case 2 of the Master Theorem, the algorithm's time complexity is $O(\log n)$. □