[CSCI 580000 Algorithms] Project 1 – Finding Peak using Divide-and-Conquer
David Xu

a. A peak $A[p]$ at index p in some array of distinct integers $A[1,2,\dots,n]$ has the largest value. A point can only be a peak if the values to the left of it are in ascending order and the values to the right of it are in descending order. Thus, a divide-and-conquer approach to solve this problem is to compare the value at the midpoint with the values directly proceeding and following it in the array. For a midpoint at index i, if the values $A[i-1,i,i+1]$ are in ascending order, then the index of the peak must be to the right of index i. Similarly, if the values $A[i-1,i,i+1]$ are in descending order, then the index of the peak must be to the left of index i. Therefore, in each divide step, the values to the left and to the right of the index can be discarded for the ascending and descending case, respectively. This occurs until the size of the array has been reduced to a length of 1, which contains the index of the peak. The base case is then when the values to the left of it are in ascending order and the values to the right of it are in descending order. The other base cases occur if the peak is at $A[1]$ and the values to the right of it are descending, or if the peak is at $A[n]$ and the values to the left of it are ascending.

b. In each step of the divide-and-conquer step, the half the elements in the array are discarded following two comparisons $A[i-1]$ with $A[i]$ and $A[i]$ with $A[i+1]$. Similarly, in the base case, there is a single comparison at linear time. Therefore, the recurrence is described as:

$$T(n) = \begin{cases} \Theta(1), & if\ n = 1, \\ T\left(\frac{n}{2}\right) + \Theta(1), & if\ n > 1. \end{cases}$$

By case 2 of the master method,

$$a = 1, b = 2, \log_b a = 0, f(n) = 1$$
$$f(n) = \Theta\left(n^{\log_b a}\right) = \Theta(n^0) = \Theta(1)$$
$$T(n) = \Theta\left(n^{\log_b a} \lg n\right) = \Theta(\lg n)$$