CSC236 tutorial exercises, Week #8

(sample solution)

1. A non-empty array A with integer entries has the property that no odd number occurs at a lower index than an even number. Devise a divide-and-conquer algorithm for finding the highest index of an even number element, or -1 if A has no elements that are even numbers. Use the Master Theorem to bound the asymptotic time complexity of your algorithm.

   Solution: The even number element with the highest index will either be followed by an odd number, or will be the last element of the array, or there are no even numbers in the array.

   ```
   # assume A is indexed from 0
   recHighestEven(A, b, e) # b and e are begin and end indices
     if b == e :
       if A[b] % 2 ==  0 : return b
       else : return b-1
     else :
       m = (b + e) // 2 # midpoint
       if A[m+1] % 2 == 1 :
         return recHighestEven(A, b, m)
       else :
         return recHighestEven(A, m+1, e)
   ```

   A recursive call on (approximately) half the array, with a constant amount of work to split and recombine the array, so the complexity can be expressed by the recurrence, if $n = |A|$:

   $$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 1 + max\{T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor)\} & \text{if } n > 1 \end{cases}$$

   In terms of the Master Theorem we have $a = 1$, $b = 2$, and $d = 0$, so the complexity is $\theta(\log n)$.

2. Consider this informal algorithm for QuickSort of a non-empty array A of distinct integers

   (a) Choose a pivot, p from A in constant time

   (b) Partition A into A_{p^-} consisting of elements less than p, $[p]$ itself, and A_{p^+} consisting of elements greater than p. Recursively QuickSort A_{p^-} and A_{p^+}

   (c) Concatenate the sorted version of A_{p^-}, $[p]$, and the sorted version of A_{p^+}

   Write a recurrence T, for the time complexity of QuickSorting A. Assume the worst (that the constant-time choice of a pivot is consistently unlucky), and use repeated substitution to find a closed form for T. Assume the best (that the constant-time choice of a pivot is consistently lucky) and use the Master Theorem to bound T.

Solution: Without more information, we don't really know the size of the partition with elements smaller than the pivot, or the partition with elements greater than the pivot. In any case, it takes proportional to the $|A| = n$ to partition A, so

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(|A_-|) + T(|A_+|) + cn \end{cases}$$

Assuming the worst would mean that the pivot is always chosen to be the largest (or smallest) element, so we only decrease the problem by 1 each recursive call:

$$\begin{aligned} T(n) &= T(n-1) + cn \\ &= T(n-2) + c(n-1) + cn \\ &\vdots \\ &= T(1) + 2c + \cdots + c(n-1) + cn \\ &= cn(n+1)/2 + 1 - c \end{aligned}$$

So the worst case is $\theta(n^2)$. Assuming the best would mean that the pivot is always chosen to split A is close to half as possible each recursive call, so

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + cn & \text{if } n > 1 \end{cases}$$

This fits the conditions of the Master Theorem with $a = 2$, $b = 2$, and $d = 1$, yielding complexity class $n \log n$. In practice, choosing the pivot randomly yields this result, on average.