CSCI 3104-Spring 2016: Assignment #2.

Assigned date: Monday 1/25/2016,
Due date: Thursday, 2/4/2016, before class
Maximum Points: 50 points (includes 5 points for legibility).

Note: This assignment *must be turned in on paper, before class.* Please do not email: it is very hard for us to keep track of email submissions. Further instructions are on the moodle page.

P1 (Divide and Conquer Maxmin, 25 points) We wish to implement an algorithm to solve the following problem:

Input: An array `a` of size `n`.

Output: `max`, `min` the maximum and minimum numbers, respectively in array `a`.

(A) Write the pseudocode for an algorithm that runs in $\Theta(n)$ time (by scanning the array from left to right).

(B) Consider a divide and conquer scheme:

1. Split the array into two equal parts.
2. Recursively compute the maximum and minimum of each part.
3. Combine the results to find maximum and minimum of the original array.

Write the pseudocode of the divide and conquer scheme using recursion. In doing so, also specify the base case.

(C) Write the recurrence relation $T(n)$ that characterizes the running time of your divide and conquer scheme.

(D) Solve $T(n)$ either by expanding or using the master method to derive a Θ bound.

Solution.
(A) The algorithm simply scans the array `a` from left to right.

```
def maxMinLinearScan(a):
    n = len(a) # compute a's length
    assert( n >= 1)
    min = a[0]
    max = a[0]
    for i in range(0,n):
        if (a[i] < min):
            min = a[i]
        if (a[i] > max):
            max = a[i]
    return (min,max)
```

(B) The recursive divide and conquer scheme is here: the initial call will be `minMaxRec(a,0, len(a))` .

```
def maxMinRec(a, l, r):
    # Compute min and max for a from indices l to r (inclusive)
    n = r - l +1 # compute a's length
    if (n == 1):

        max = a[l]
        min = a[r]

    elif (n == 2):

        if (a[l] <= a[r]):
            min = a[l]
            max = a[r]
        else:
            min = a[r]
            max = a[l]
    else:

        mid = (l+r)/2    # Compute the mid point
        # Recursive call for two parts
        (min1, max1) = maxMinRec(a, l, mid)
        (min2, max2) = maxMinRec(a, mid, r)
        # Combine the results of the call
        if (min1 <= min2):
            min = min1
        else:
            min = min2
        if (max1 <= max2):
            max = max2
        else:
            max = max1

    return (min,max)
```

(C) The recurrence for running time is

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + C_1 & n > 2 \\ C_0 & n \leq 2 \end{cases}$$

(D) Solving using case-1 of master method: $b = 2, a = 2$. We have $\alpha = \log_b(a) = 1$. Also $f(n) = C_1 = O(n^\alpha)$. Therefore, we conclude that case-1 of the master method applies. The solution is $T(n) = \Theta(n)$.

▲

P2 (Divide and Conquer Majority, 25 points) An element `e` is a majority element in an array

`a` of size `n` if and only if it occurs $\lceil \frac{n}{2} \rceil + 1$ or more times.

We wish to implement an algorithm to solve the following problem:

Input: An array `a` of size `n`.

Output: A number `e` if it is the majority element of array `a` and NONE if `a` has no majority element.

We use the following divide and conquer scheme:

1. Split `a` into two equal parts.
2. Recursively find whether a majority element exists for each part or NONE.
3. Combine the results of the recursive calls.

(A) Suppose array `a` of size n is divided into two parts `a1` and `a2`. Let `e1` and `e2` be the majority elements of `a1` and `a2`, respectively. What is the majority element of the overall array `a`?

(B) Write the pseudocode for the divide and conquer algorithm above. Pay special attention to the base case.

(C) Derive and solve the recurrence for the running time $T(n)$.

Solution.

(A) Suppose array `a` of size n is divided into two parts `a1` and `a2`. Let `e1` and `e2` be the majorities of `a1` and `a2`.

If `e1` == `e2` then the majority of `a` exists and is `e1`

If `e1` != `e2` then the majority is either `e1` or `e2`. We simply count whether `e1` or `e2` is the majority.

(Check that `e1`, `e2` need not be numbers, i.e this works even if `e1` = NONE or `e2` = NONE).

(B) Assume function `countOccurrences(a, e)` returns the count of how many times element `e` occurs in array `a`. It runs using a single for loop in $\Theta(n)$ time where n is the length of the array `a`.

```
    def findMajority(a):
        n = len(a) # compute a's length
        if (n == 1):
                maj = a[0]   # If it is 1, then the majority is a[0]
        else:
                mid = int(n/2)   # Take the mid point
                (maj1) = findMajority(a[0:mid])
# Compute majority on the left
                (maj2) = findMajority(a[mid:n])
# Compute majority on the right

                if (maj1 == maj2): # If the answers from left and right are t
                        maj = maj1
                else:   # Otherwise
```

```
                    if (maj1 != None):
                        cnt1 = countOccurences(a, maj1) # Count how many t
                    else:
                        cnt1 = 0

                    if (maj2 != None):
                        cnt2 = countOccurences(a, maj2) # Count how many t
                    else:
                        cnt2 = 0


                    if (cnt1 > n/2): # If maj1 is the majority
                         maj = maj1
                    elif (cnt2 > n/2): # else if maj2 is the majority
                         maj = maj2
                    else: # Otherwise, there is no majority
                         maj = None
        return maj
```

(C) Running time recurrence :

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + C_1 n & n > 2 \\ C_0 & n \leq 2 \end{cases}$$

Solution is $\Theta(n \log(n))$

▲