CSE 101 Practice Midterm Questions: SOLUTIONS

For each problem, an example of a minimal acceptable answer is given, to give you an idea of how much detail is needed on the midterm. A more detailed explanation follows this for some problems. In some cases, there are different answers that would also be acceptable.

1. **(a)** 2. Explanation: This algorithm finds the first row whose elements add up to $2n$. It returns NULL if there is no such row. L(A,n) returns 2, since the second row's elements sum to $6 = 2n$.

   (b) $\Theta(n^2)$. Explanation: The outer **for** loop goes through at most n iterations. The inner **for** loop always goes through n values of j, taking constant time in lines 4-6 each iteration. So the inner loop uses a total of $\Theta(n)$ time to complete its execution. In the worst case, when the outer loop iterates n times, executing this complete inner loop each time, total time is $\Theta(n^2)$.

   (c) $\Theta(n)$. Explanation: In the best case, the very first row sums to $2n$, so that we return before trying any other values of i. We only execute the inner loop once, for a total of $\Theta(n)$ time.

2. MODE(A)

```
1   MERGE-SORT(A,1,length[A])
2   c ← A[1]
3   d ← 1
4   m ← 0
5   for i ← 2 to length[A]
6       do if A[i] = c
7           then d ← d + 1
8               if d > m
9                   then m ← d
10                      l ← c
11          else c ← A[i]
12              d ← 1
13  return l
```

   Explanation: We can find the mode by sorting the sequence: then, all duplicates will be next to each other, so that we can easily count how many times each element occurs. In the algorithm above, c holds the current element for which we are counting number of occurrences, d holds the number of occurrences of this element so far, m holds the maximum number of occurrences of any element so far, and l holds this maximally-occurring element.

   Line 1, MERGE-SORT, takes $\Theta(n \lg n)$ time, and the remainder of the algorithm loops through the elements once, spending constant time on each, for a total of $O(n)$ time. So, total time is $O(n \lg n)$ as required.

3. Recurrence:

$$T(n) = 3T(\frac{n}{3}) + \Theta(n)$$

   Using the master method, case 2 applies, since $\Theta(n^{\log_b a}) = \Theta(n^1)$, which is the same as $f(n)$ here. The solution given by case 2 is:

$$T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(n \lg n)$$

   Explanation: Let $T(n)$ be the time to sort a sequence of length n. Lines 1-2 take constant time. Each of lines 3-5 is a recursive call on exactly $\frac{n}{3}$ elements (this is always possible since n is assumed to be a power of 3). Lines 6-7 take $\Theta(n)$ time, as shown in class. This gives the recurrence:

$$T(n) = 3T(\frac{n}{3}) + \Theta(n)$$

The master theorem is then applied with $a = 3$, $b = 3$, and $f(n) = \Theta(n)$. Since $\log_b a = \log_3 3 = 1$, $f(n) = \Theta(n^{\log_b a})$, so we are in case 2 of the master theorem, which tells us:

$$T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(n \lg n)$$

4. Prove RANDOMIZED-SELECT correct by induction on $n = r - p + 1$, the number of elements from which we are selecting. For the base case of $n = 1$, the check on line 1 passes, so line 2 immediately returns the single element. This is correct since it is the only element available.

   Now assume that RANDOMIZED-SELECT is correct for less than n elements (this is the induction hypothesis), and show it is correct for n elements (with $n > 1$ now). Line 1's check fails, so execution continues on line 3. By the correctness of RANDOMIZED-PARTITION, $A[p \ldots q]$ holds the smallest k elements of $A[p \ldots r]$ after line 3, and $A[q+1 \ldots r]$ holds the $n - k$ largest elements. Since the partition is nontrivial, $1 \leq k \leq n - 1$, so the recursive calls on lines 6-7 are on fewer than n elements and therefore return correct values by the induction hypothesis. We are looking for the i'th smallest element of $A[p \ldots r]$. So, if $i \leq k$ then this is the i'th smallest element among the k smallest elements of $A[p \ldots r]$: in this case, the recursive call on line 6 returns this value. Otherwise, the i'th smallest element is the $i - k$'th smallest among the $n - k$ largest elements of $A[p \ldots r]$: line 7 correctly returns this value in this case.

   By induction, RANDOMIZED-SELECT is correct for all values of n.

5. Insert the following line after line 3, within the **for** loop that starts on line 2:

   $a[i, i] \leftarrow 1$

   Insert the following line between lines 11 and 12:

   $$a[i, j] \leftarrow a[i, k] * a[k + 1, j]$$

   Insert the following two lines between lines 12 and 13; the **else** here is attached to the **if** on line 10:

   else if $q = m[i, j]$
       then $a[i, j] \leftarrow a[i, j] + a[i, k] * a[k + 1, j]$

   Finally, change line 13 to be:

   13 **return** m and s and a

   Explanation: Initialize each $a[i, i]$ to one since there is only one way to parenthesize a single matrix. When a new minimum number of multiplications is found, initialize the corresponding a entry with the total number of ways to combine a minimal parenthesization on the left side of the split with a minimal parenthesization on the right side. The total number of ways to combine these is the product of the numbers of optimal parenthesizations on either side of the split. When a different split yields the same number of multiplications as the minimum, increase the corresponding a count by this product of the numbers on either side of the split.