CS 325 - Winter 2020
Homework #2

Problem 1.

Problem 1.a. (2 points)

- $T(n) = b \cdot T(n-1) + 1$ where b is a fixed positive integer greater than 1.

Solution:

- $T(n) = b \cdot T(n-1) + 1$ where $T(n-1) = b \cdot T(n-2) + 1$
- $T(n) = b \cdot (b \cdot T(n-2) + 1) + 1 = b^2 \cdot T(n-2) + b + 1$ where $T(n-2) = b \cdot T(n-3) + 1$
- $T(n) = b \cdot (b \cdot (b \cdot T(n-3) + 1) + 1) + 1 = b^3 \cdot T(n-3) + b^2 + b + 1$
- $\cdots$
- $T(n) = b^{n-1} \cdot T(1) + b^{n-2} + \cdots + b^2 + b + 1 \leq \Sigma_{i=0}^{n} b^i$ assuming $T(1)$ is constant.
- $T(n) \leq (1 - b^{n+1})/(1-b)$ if $b > 1$
- Thus, we have $T(n) = O(b^n)$.

Problem 1.b. (2 points)

- $T(n) = 3 \cdot T(n/9) + n \cdot \log n$

Solution:

Create the recursion tree. The total cost of the tree will be $\Sigma_{i=0}^{(\log_9 n)-1}[(n/3^i) \cdot \log(n/9^i)] = \Sigma_{i=0}^{(\log_9 n)-1}[(1/3^i) \cdot (n \cdot \log n)] - \log 9 \cdot (\Sigma_{i=0}^{(\log_9 n)-1} i) \cdot n \quad \leq 3/2 \cdot n \cdot \log n = O(n \cdot \log n)$

Problem 2. (6 points)
Solve Exercise 4.1-5 from the 3rd Ed. of the textbook.

Solution. The logic of the solution could be something similar to the following (assuming the array A[1 .. n]):

```
Sum := ¬∞
low, high := unassigned;
tmpSum := 0;
tmpLow := 1;
for i := 1 to A[n] {
    add A[i] to the tmpSum;
    If (tmpSum > Sum) then
```

```
            Sum := tmpSum
            low := tmpLow
            high := i
        If tmpSum < 0 then tmpSum := 0
                tmpLow := i +1
    }
    return(low, high, Sum);
```

Problem 3.
Consider the recurrence $T(n) = 3 \cdot T(n/2) + n$.

Problem 3.a. (3 points)

- Solution

Create the recursion tree. The total cost of the tree will be at most $n \cdot \Sigma_{i=0}^{(\log n)} (3/2)^i$, where $\Sigma_{i=0}^{(\log n)} (3/2)^i = [(3/2)^{1+\log n} - 1]/((3/2) - 1) = 2 \cdot [(3/2)^{1+\log n} - 1]$

The Σ is the summation of a geometric series, which will evaluate to $3 \cdot (n^{\log 3}/n) - 2$. Multiplying it by n would give us $3 \cdot n^{\log 3} - 2n$, which is $O(n^{\log 3})$.

Problem 3.b. (3 points)

- Solution

- *Base Case*: Assuming $T(1) = 1$, we have $T(2) = 5$. Now, we have $T(2) \leq c \cdot 2^{\log 3}$.
That is, $5 \leq c \cdot 2^{\log 3}$ holds for $c > 1.7$.
- *Induction Hypothesis*: For all $m < n$, $T(m) \leq c \cdot m^{\log 3} - d \cdot m$ holds for some $d > 0$.
- *Induction Step*: Start with $T(n) = 3T(n/2) + n$ and apply the hypothesis to get
$T(n) \leq 3 \cdot [c \cdot (n/2)^{\log 3} - d \cdot (n/2)] + n$
$T(n) \leq 3 \cdot c \cdot (n/2)^{\log 3} - 3d \cdot (n/2) + n = c \cdot n^{\log 3} - (3d/2 - 1) \cdot n$
$T(n) \leq c \cdot n^{\log 3} - (3d/2 - 1) \cdot n$ for $d > 2/3$.

Problem 4.

Problem 4.a. (3 points)

- For $\alpha \leq 1/2$, badSort initially sorts $A[0 \cdots m-1]$, which is at most equal to the left half sub-array. Then, it sorts some elements in the second half of the array, and finally sorts $A[0 \cdots m-1]$ again. Since the two sub-arrays do not overlap, there may be an element in the left sub-array that is greater than some elements in the right sub-array, which results in an unsorted array.

Problem 4.b. (2 points)

- One issue with $\alpha = 3/4$ is that for $n = 3$, we have $m = \lceil 3/4 \cdot n \rceil = 3$. This means that badSorts falls in a never-ending loop. To fix this issue, the termination condition should be $(n = 3)$ instead of $(n = 2)$. Moreover, instead of the swap operation, we can use insertionSort to sort an array of 3 elements in constant time.

Problem 4.c. (2 points)

- $T(n) = 3T(\alpha \cdot n) + \theta(1)$ because badSort recurses on three sub-problems of size $\alpha \cdot n$ plus the constant cost of the terminating case (i.e., swapping $A[0]$ and $A[1]$).

Problem 4.d. (2 points)

- Use Master theorem for $a = 3$ and $b = 1/\alpha$, where we have $\log_{1/\alpha} 3$. For $\alpha = 2/3$, we have $\log_{3/2} 3 = 2.71$. Thus, $f(n) = c = O(n^{2.71})$. Therefore, $T(n) = \theta(n^{2.71})$.

Problem 5.

Problem 5.a. (3 points)

- **Implementation**: Code should compile and execute correctly for $\alpha = 2/3$. (3 points)

Problem 5.b. (3 points)

- **Modify code**: (1) Random array generation (1 point).
  (2) Generation of timing data: Time table for $\alpha = 2/3$ and $\alpha = 3/4$. (1 point for the timing of $\alpha = 2/3$ and 1 point for $\alpha = 3/4$)

Problem 5.c. (2 points)

- **Plot data and fit a curve**: (1) Plot data for $\alpha = 2/3$. Polynomial curves with the degree $2 \leq d \leq 3$ receive full credit. (1 point).
  (2) Plot data for $\alpha = 3/4$. Polynomial curves with the degree $3 \leq d \leq 4$ receive full credit. (1 point)
  If the degree of the curve does not quite match the aformentioned curves, then give 1 point out of 2.

Problem 5.d. (2 points)

- **Comparison**: $\alpha = 2/3$ gives better results. Give 2 points if this is mentioned. For other explanations that may not be fully correct, give 1 point.

Submit a copy of all your code files and a README file that explains how to compile and run your code in a ZIP file to TEACH. We will only test execution with an input file named data.txt.