CS 141, Spring 2019 Homework 2

Problem 1. (25 points)
Given the following recurrence relation

$$T(n) = \begin{cases} 1 & n = 1 \\ T\left(\frac{n}{9}\right) + \sqrt{n} & n > 1 \end{cases}$$

1. Solve it exactly (i.e., without using any asymptotic notation) by iterative substitutions
2. Prove by induction that your exact solution is correct (do not prove a bound, but the exact solution)

Answer: We have

$$\begin{aligned} T(n) &= T(n/9) + \sqrt{n} \\ &= T(n/9^2) + \sqrt{n}\,(1/3 + 1) \\ &= T(n/9^3) + \sqrt{n}\,(1/9 + 1/3 + 1) \\ &\dots \\ &= T(n/9^i) + \sqrt{n}\left(1/3^{i-1} + 1/3^{i-2} + \ldots + 1/3^1 + 1/3^0\right) \\ &= T(n/9^i) + 3/2\sqrt{n}\left(1 - 1/3^i\right) \end{aligned}$$

now we set $n/9^i = 1$ which is $i = \log_9 n$ and we get

$$\begin{aligned} T(n) &= T(1) + 3/2\sqrt{n}\left(1 - 1/3^{\log_9 n}\right) \\ &= 1 + 3/2\sqrt{n}\,(1 - 1/\sqrt{n}) \\ &= \frac{3\sqrt{n} - 1}{2} \end{aligned}$$

We now prove by induction that $T(n) = \frac{3\sqrt{n}-1}{2}$ is the correct solution of the recurrence relation.
Base case ($n = 1$). $T(1) = \frac{3\sqrt{1}-1}{2} = 1$.
Induction step. Assume the statement true for $n/9$, that is

$$T\left(\frac{n}{9}\right) = \frac{3\sqrt{n/9} - 1}{2} = \frac{\sqrt{n} - 1}{2}$$

We have

$$\begin{aligned} T(n) &= T\left(\frac{n}{9}\right) + \sqrt{n} \\ &= \frac{\sqrt{n} - 1}{2} + \sqrt{n} \\ &= \frac{3\sqrt{n} - 1}{2} \end{aligned}$$

Problem 2. (25 points)

Using the Master method, give an asymptotic tight bound for $T(n)$ in the following recurrence relation

$$T(n) = \begin{cases} 1 & n = 1 \\ T\left(\frac{n}{3}\right) + n\log_3 n & n > 1 \end{cases}$$

Answer: Case 3 of Master theorem applies. First note that $n^{\log_b a} = n^{\log_3 1} = n^0 = 1$. The first condition for case 3 is $n\log_3 n \in \Omega(n^\epsilon)$ which is satified for $\epsilon = 1$. The second condition is $af(n/b) \leq \delta f(n)$, which translates to

$$\begin{aligned} (n/3)\log_3(n/3) &= (n/3)\log_3 n - (n/3)\log_3 3 \\ &= (n/3)\log_3 n - (n/3) \\ &\leq \delta n \log_3 n \end{aligned}$$

The last inequality is satisfied by $\delta = 1/3 < 1$.
The conclusion is $T(n) = \Theta(n\log n)$.

Problem 3. (25 points)

Suppose that we have designed three divide-and-conquer algorithms that solve a particular problem, where the input size is n. The first one solves four subproblems of size $n/2$ and the cost of combining the solutions of the subproblems to obtain a solution for the original problem is n^2. The second solves three subproblems of size $n/2$ and requires $n^2\sqrt{n}$ time for combining the solutions. The third solves five subproblems of size $n/2$ and requires $n\log n$ time for combining the solutions. Assume that all three take $\Theta(1)$ when $n = 1$. Which algorithm would you choose and why? Show your work using the Master method.

Answer: We have

$$T_1(n) = \begin{cases} 1 & n = 1 \\ 4T_1(n/2) + n^2 & n > 1 \end{cases}$$

and

$$T_2(n) = \begin{cases} 1 & n = 1 \\ 3T_2(n/2) + n^2\sqrt{n} & n > 1 \end{cases}$$

and

$$T_3(n) = \begin{cases} 1 & n = 1 \\ 5T_3(n/2) + n\log n & n > 1 \end{cases}$$

The first one is case II of the Master theorem, because $n^2 \in \Theta(n^{\log_2 4}\log^k n)$ for $k = 0$. The solution is $T_1(n) \in \Theta(n^2 \log n)$.

The second one is case III, because $n^2\sqrt{n} \in \Omega(n^{\log_2 3+\epsilon})$ for $\epsilon = 2.5 - \log_2 3$ which is positive. Also, we have to check whether $3(n/2)^{2.5} \leq \delta n^{2.5}$. The inequality is satified by $\delta = 3/(4\sqrt{2}) < 1$. Therefore $T_2(n) \in \Theta(n^2\sqrt{n})$.

The third recurrence relation is case I of the Master Theorem, because $n \log n \in O(n^{\log_2 5 - \epsilon})$ for $\epsilon = \log_2 5 - 2$ which is about 0.3219. In this case we are upper bounding $n \log n$ with n^2 which holds. The solution is $T_3(n) \in \Theta(n^{\log_2 5})$ where $\log_2 5 \approx 2.3219$.

We should choose the first algorithm because both $n^{2.5}$ (second algorithm) and $n^{2.3}$ (third algorithm) grow asymptotically faster than $n^2 \log n$.

Problem 4. (25 points)

The *median* of a set of numbers $\{a_1, a_2, \ldots, a_n\}$ is the element a_i such that there are $\lceil n/2 \rceil$ elements smaller than or equal to a_i, and there are $\lfloor n/2 \rfloor$ greater than or equal to a_i. In other words, the median is the element in the middle when the elements are sorted. For example, the median of $\{7, 3, 4, 1, 9, 2, 13\}$ is 4.

You are given two sorted arrays A and B of size n each (for simplicity, you can assume n to be some power of 2 and that the numbers are distinct). Give an algorithm to find the median of all $2n$ numbers in $O(\log n)$ time.

Answer: The strategy is divide and conquer. First find the median m_A and m_B of array A and B in constant time. Compare m_A and m_B. If $m_A > m_B$, then the median of the two arrays cannot be in the second half of A (those elements larger than m_A), and it cannot be in the first half of B (those elements are smaller than m_B). We therefore throw away the second half of A and the first half of B and recursively search for the median in the rest of the arrays. In the same manner, if $m_A \leq m_B$, we throw away the first half of A (those smaller than m_A) and the second half of B (those larger than m_B). The recursion ends when each array has two elements and we can computes the median of those 4 elements in constant time.

Since each call divides the array in two, and constant work is done for each division, the recurrence relation is $T(n) = T(n/2) + c$ which has solution $O(\log n)$.