CSI - 3105 Design & Analysis of Algorithms
Fall 2020

Assignment #1

Deadline: Thursday, October 1st, 2020 before 10:00 AM (to be delivered in BrightSpace)

Note: Make sure your handwriting is readable, otherwise your assignment will not be marked.

1. (10 marks) Assume that $n = 5^k$, where k is a positive integer, and solve the following recurrence by unfolding.

$$T(n) = \begin{cases} 2 & \text{if } n = 1, \\ 10\,T\left(\frac{1}{5}n\right) + \frac{1}{2}n & \text{if } n \geq 2. \end{cases}$$

Your final answer must be an exact formula in terms of n (do not use O-notation). Your final answer must be simplified as much as possible. You must provide all details of your solution.

Solution: Assuming that $n = 5^k$, we have

$$\begin{aligned}
T(n) &= 10\,T\left(\frac{1}{5}n\right) + \frac{1}{2}n \\
&= 10\left(10\,T\left(\frac{1}{5^2}n\right) + \frac{1}{2}\cdot\frac{n}{5}\right) + \frac{1}{2}n \\
&= 10^2\,T\left(\frac{1}{5^2}n\right) + 2\cdot\frac{1}{2}n + \frac{1}{2}n \\
&= 10^2\left(10\,T\left(\frac{1}{5^3}n\right) + \frac{1}{2}\cdot\frac{n}{5^2}\right) + 2\cdot\frac{1}{2}n + \frac{1}{2}n \\
&= 10^3\,T\left(\frac{1}{5^3}n\right) + 2^2\cdot\frac{1}{2}n + 2\cdot\frac{1}{2}n + \frac{1}{2}n \\
&= 10^3\left(10\,T\left(\frac{1}{5^4}n\right) + \frac{1}{2}\cdot\frac{n}{5^3}\right) + 2^2\cdot\frac{1}{2}n + 2\cdot\frac{1}{2}n + \frac{1}{2}n \\
&= 10^4\,T\left(\frac{1}{5^4}n\right) + 2^3\cdot\frac{1}{2}n + 2^2\cdot\frac{1}{2}n + 2\cdot\frac{1}{2}n + \frac{1}{2}n \\
&= 10^4\,T\left(\frac{1}{5^4}n\right) + \left(2^3 + 2^2 + 2^1 + 2^0\right)\cdot\frac{1}{2}n \\
&\vdots
\end{aligned}$$

$$
\begin{aligned}
&= 10^k\, T\left(\frac{1}{5^k}n\right) + \frac{1}{2}n \sum_{i=0}^{k-1} 2^i \\
&= 10^k\, T\left(\frac{1}{5^k}n\right) + \frac{1}{2}n\left(2^k - 1\right) && \text{since } \sum_{i=0}^{k-1} 2^i = 2^k - 1, \\
&= 2 \cdot 10^k + \frac{1}{2}n\left(2^k - 1\right) && \text{since } T\left(\frac{1}{5^k}n\right) = T(1) = 2, \\
&= 2 \cdot 2^k \cdot 5^k + \frac{1}{2}n\left(2^k - 1\right) \\
&= 2n \cdot 2^k + \frac{1}{2}n\left(2^k - 1\right) && \text{since } n = 5^k, \\
&= \frac{5}{2}n \cdot 2^k - \frac{n}{2} \\
&= \frac{5}{2}n\left(5^{\log_5(2)}\right)^k - \frac{n}{2} \\
&= \frac{5}{2}n\left(5^k\right)^{\log_5(2)} - \frac{n}{2} \\
&= \frac{5}{2}n \cdot n^{\log_5(2)} - \frac{n}{2} && \text{since } n = 5^k, \\
&= \frac{5}{2}n^{1+\log_5(2)} - \frac{n}{2} \\
&= \frac{5}{2}n^{\log_5(10)} - \frac{n}{2}.
\end{aligned}
$$

2. (10 marks) Design a deterministic algorithm to solve the following problem.

   input: An array $A[1..n]$ of n integers.

   output: Two different indices i and j such that $A[i] + A[j] = 2020$, if such indices exist. Otherwise, return **NONE**.

   Your algorithm must take $O(n \log(n))$ time. You must describe your algorithm in plain English (no pseudocode) and you must explain why the running time of your algorithm is $O(n \log(n))$.

   Solution: Be careful: we need to return the indices i and j, not the numbers $A[i]$ and $A[j]$. Moreover, when we sort A, we destroy the original indices.

   So first, scan A and replace each number $x \in A$ by a pair (x, k), where k is the index of x in A. To do that, we have to visit each position in A once. This takes $O(n)$ time. For instance, $\{17, 2, -140, 2028, 42, -8\}$ becomes $\{(17,1), (2,2), (-140,3), (2028,4), (42,5), (-8,6)\}$.

   Second, sort A (with respect to the first number in each pair) using Merge sort. This takes $O(n \log(n))$ time. For instance, $\{(17,1), (2,2), (-140,3), (2028,4), (42,5), (-8,6)\}$ becomes $\{(-140,3), (-8,6), (2,2), (17,1), (42,5), (2028,4)\}$.

Finally, we scan A and for each number $x \in A$, we want to know whether $2020 - x$ is in A. We can search for $2020 - x$ in A by doing binary search. If we find $2020 - x$ in A (with an index different than that of x), we return its index, together with the index of x and we stop. If all binary searches were unsuccessful, we return **NONE** and we stop. Binary search takes $O(\log(n))$ time and we run it at most once for each of the n numbers in A. This makes $O(n \log(n))$ time for this part of the algorithm. In our example, we first consider the pair $(-140, 3)$. We do binary search with $2020 - (-140) = 2140$, which is unsuccessful. Then, we get to the pair $(-8, 6)$. We do binary search with $2020 - (-8) = 2028$ and we find it! So we return 4 and 6.

In total, this algorithm takes $O(n) + O(n \log(n)) + O(n \log(n)) = O(n \log(n))$ time.