CSE 3500 Algorithms and Complexity

Fall 2016; Exam I; Solutions

1. (a) **FALSE.** Here is a counterexample: Let $f(n) = 3n$ and $g(n) = 2n, u(n) = n + 1$, and $v(n) = n$. Clearly, $f(n) = \Theta(g(n))$ and $u(n) = \Theta(v(n))$. $f(n) - g(n) = n$ and $u(n) - v(n) = 1$. Clearly, $n \neq \Theta(1)$.

   (b) **TRUE.** Note that $2^{\log n} = n$ and hence the RHS is $\Theta(n^2 \log n)$. LHS=$\Theta(n^2 \log n)$ and therefore the given statement is true.

2. Consider the following algorithm:

   Pick a random sample S of k elements from X; Find and output the maximum element of S. (The value of k will be fixed in the analysis.)

   Analysis:

   The output of the above algorithm will be incorrect only if all the keys in the random sample come from the smallest 90% of the elements of X. Probability that a randomly picked element comes from the smallest 90% of the elements of X is ≤ 0.9. Thus, the probability that the algorithm outputs an incorrect answer is $\leq (0.9)^k$. We want this probability to be $\leq n^{-\alpha}$. This happens when $k \geq \frac{\alpha \log n}{\log(1/0.9)}$. This implies that the run time of the algorithm is $O(\log n)$.

3. Here is a linear time algorithm for the given problem:

   Compute $sum = \sum_{j=1}^{l} k_j$;
   if $sum = s$ **then** output 1 and quit;
   for $i = 2$ **to** $(n - l + 1)$ **do**
       $sum = sum - k_{i-1} + k_{i+l-1}$;
       if $sum = s$ **then** output i and quit;
   Output no;

   For $i = 1$ we spend $O(l)$ time and for each value of $i = 2, 3, \ldots, (n - l + 1)$ we spend $O(1)$ time. Thus the total time is $O(n)$.

4. Keep two 2-3 trees N and S. We also keep a variable MS to store the maximum salary value. In N store all the records with the name as the key for each record and in S

store all the records with the social security number as the key for each record. To process $\mathsf{Insert}(Name, SSN, salary)$, we insert $Name$ into N. In the same node we also store SSN. Also, we insert SSN into the tree S. In the same node we store $Name$. In addition, we set $MS = \max\{MS, salary\}$. To process $\mathsf{Find\_Name}(SSN)$, we search for a record whose key is SSN in the tree S. The name in this node will be output. The run time is $O(\log n)$. We process $\mathsf{Find\_SSN}(Name)$ in a similar manner. To process $\mathsf{MaxSalary}()$ we return the value of MS.

5. (a) $T(n) = 125\,T\left(\frac{n}{5}\right) + n^2$. Here $a = 125, b = 5, n^{\log_b a} = n^3, f(n) = n^2$. Case 1 of the Master theorem applies. Therefore, $T(n) = \Theta(n^3)$.

   (b) $T(n) = T\left(\frac{n}{16}\right) + T\left(\frac{n}{25}\right) + T\left(\frac{n}{400}\right) + \sqrt{n}$.
   We claim that $T(n) = O(\sqrt{n})$. This can be proven by induction.
   Hypothesis: $T(n) \le c\sqrt{n}$, for some constant c.
   Base case: easy.
   Induction step: Assume the hypothesis for all the inputs of size up to $n-1$. We'll prove it for inputs of size n.

$$T(n) = T\left(\frac{n}{16}\right) + T\left(\frac{n}{25}\right) + T\left(\frac{n}{400}\right) + \sqrt{n} \le c\sqrt{n/16} + c\sqrt{n/25} + c\sqrt{n/400} + \sqrt{n}$$
$$= \frac{c}{2}\sqrt{n} + \sqrt{n}.$$

   RHS will be $\le c\sqrt{n}$ if $c \ge 2$. As a result, it follows that $T(n) \le 2\sqrt{n} = O(\sqrt{n})$. $\square$

6. We first sort $a_1, a_2, \ldots, a_q$ to order the intervals. This takes $O(q \log q)$ time. Let $Y = [a'_1, b'_1], [a'_2, b'_2], \ldots, [a'_q, b'_q]$ be the ordered sequence of intervals.

   for $i = 1$ **to** q **do**
       $n_i = 0$;
   for $i = 1$ **to** n **do**
       Perform a binary search for k_i in Y;
       Let $[a_j, b_j]$ be the interval that k_i belongs to;
       $n_j = n_j + 1$;
   for $i = 1$ **to** q **do**
       Output n_i;

To sort the intervals it takes $O(q \log q)$ time. For each input key we perform a binary search that takes $O(\log q)$ time. Thus the total run time is $O(q \log q + n \log q) = O(n \log q)$. $\square$