CS3230: Assignment for Week 3 Solutions

Due: Sunday, 6th Feb 2022, 11:59 pm SGT.

---

1. (a) Using the master method, $a = 3$, $b = 2$, $f(n) = n^2$. We have $n^{\log_b a} = n^{\log_2 3}$. Since $\log_2 3 < 2$, we are in Case 3 of the master method. The regularity condition is satisfied because $af(n/b) = 3f(n/2) = 3n^2/4 \leq cf(n)$ for $c = 3/4$. So $T(n) = \Theta(f(n)) = \Theta(n^2)$.

   (b) Using the recursion tree, the amount of work we have to do at each level is $\Theta(n)$, and the number of levels is $\Theta(\lg n)$ (in particular, it is upper-bounded by $\log_{4/3} n$ and lower-bounded by $\log_4 n$), so $T(n) = \Theta(n \lg n)$.

   (c) We claim that $T(n) = \Theta(n)$. Since $T(n) \geq n$, it suffices to show that $T(n) = O(n)$. We do so using the substitution method. Choose a sufficiently large constant c so that $T(n) \leq cn$ for $n \leq 4$ and $\left(\frac{2c}{c-2}\right)^2 \leq 5$; since $\lim_{c\to\infty} \frac{2c}{c-2} = 2$, this is possible. We prove by strong induction that $T(n) \leq cn$ for all n; the base case $n \leq 4$ holds by our choice of c. For the inductive step, assuming that the statement holds up to $n-1$, we have

$$\begin{aligned} T(n) &= T(n/2) + T(\sqrt{n}) + n \\ &\leq \frac{cn}{2} + c\sqrt{n} + n \\ &\leq cn, \end{aligned}$$

   where for the last inequality we use the assumption that $\frac{2c}{c-2} \leq \sqrt{5} \leq \sqrt{n}$. This completes the induction.

2. When the algorithm merges two sorted subarrays A and B of size t each, the minimum number of comparisons it makes is t—this happens when $A_t < B_1$ or $B_t < A_1$ (e.g., if the first array is $1, 2, \ldots, t$ and the second array is $t+1, t+2, \ldots, 2t$). The maximum number of comparisons it makes is $2t-1$—this happens when $A_t > B_{t-1}$ and $B_t > A_{t-1}$ (e.g., if the first array is $1, 3, \ldots, 2t-1$ and the second array is $2, 4, \ldots, 2t$).

   If the original array is $1, 2, \ldots, n$, then the minimum number of comparisons is achieved for every merge. Merging two subarrays of size $n/2$ requires $n/2$ comparisons, merging two subarrays of size $n/4$ requires $n/4$ comparisons and this is done twice, and so on. So the total number of comparisons is

$$(n/2)\cdot 1 + (n/4)\cdot 2 + (n/8)\cdot 4 + \cdots + 1\cdot(n/2) = \overbrace{n/2 + n/2 + \cdots + n/2}^{\lg n \text{ times}} = \frac{n\lg n}{2}.$$

For the maximum, we claim that there is an original array consisting of $1, 2, \ldots, n$ in some order such that the last number is n and the maximum number of comparisons is achieved for every merge. To see this, we proceed by induction on n. For the base case $n = 2$, the array $1, 2$ requires 1 comparison, which is already the maximum. Suppose that A is such an array of size n. To construct an array B of size $2n$, we let $B_t = 2A_t - 1$ and $B_{n+t} = 2A_t$ for all $t = 1, 2, \ldots, n$. (For example, the array for $n = 4$ is $1, 3, 2, 4$.) By the induction hypothesis, B consists of $1, 2, \ldots, 2n$ in some order, $B_n = 2n - 1$, and $B_{2n} = 2n$. Since the relative size of the numbers within each half of B is the same as that in A, by the induction hypothesis, every merging of two subarrays of size less than n takes the maximum number of comparisons. Moreover, since $B_n > B_{2n-1}$ and $B_{2n} > B_{n-1}$, merging the two subarrays of size n takes the maximum number of comparisons, $2n - 1$. This completes the induction. The total number of comparisons required for sorting this array is

$$\begin{aligned}
&(2(n/2) - 1)\cdot 1 + (2(n/4) - 1)\cdot 2 + (2(n/8) - 1)\cdot 4 + \cdots + (2(1) - 1)\cdot(n/2) \\
&= (\overbrace{n + n + \cdots + n}^{\lg n \text{ times}}) - (1 + 2 + 4 + \cdots + (n/2)) \\
&= n\lg n - (n - 1) \\
&= n\lg n - n + 1.
\end{aligned}$$

Note that this bound matches the claim made in Lecture 1.

3. We are interested in finding a local minimum of the array A. (Note that a local minimum always exists; for example, a global minimum is also a local minimum.)

   We use a divide-and-conquer approach. If $n = 1$, return A_1. If $n = 2$, make one comparison and return the smaller number. Assume that $n \geq 3$. Consider $A_{\lceil n/2 \rceil}$, and compare it with both of its neighbors. If it is less than both of its neighbors, return it. Else, suppose it is greater than the neighbor to its left, $A_{\lceil n/2 \rceil - 1}$ (we proceed similarly if it is greater than the neighbor to its right). We claim that any local minimum in the subarray consisting of the first $\lceil n/2 \rceil - 1$ elements is also a local minimum in the entire array. The claim is obvious for $A_1, A_2, \ldots, A_{\lceil n/2 \rceil - 2}$, while for $A_{\lceil n/2 \rceil - 1}$, it is true because $A_{\lceil n/2 \rceil - 1} < A_{\lceil n/2 \rceil}$. So we may recurse on this subarray. The number of comparisons made satisfies $T(n) = T(n/2) + 2$, which (e.g., by the master method) gives $T(n) = \Theta(\lg n)$.