CIS 606 Analysis of Algorithms

Spring Semester, 2025

Bonus Assignment (100 pts)

Due Date: 11:59 p.m., Friday, May 2, 2025.

Note: If not specified, the base of log is 2. This applies to all assignments of this course.

1. **(10 points)** Order the following list of functions in increasing order asymptotically (i.e., from small to large, as we did in class).

$$\begin{array}{cccccc} \log n & n! & 2^{500} & 2^n & \log(\log n)^2 & 2^{\log n} \\ \log^3 n & n\log n & \log_4 n & n^3 & \sqrt{n} & n^2\log^5 n \end{array}$$

2. **(10 points)** Solve the following recurrences (you may use any of the methods we studied in class). Make your bounds as small as possible (in the big-O notation). For each recurrence, $T(n)$ is constant for $n \leq 2$.

   (a) $T(n) = 4 \cdot T(\frac{n}{2}) + n\sqrt{n}$.

   (b) $T(n) = 2 \cdot T(\frac{n}{2}) + n\log n$.

3. **(30 points)** Let $A[1 \cdots n]$ be an array of n *distinct* numbers (i.e., no two numbers are equal). If $i < j$ and $A[i] > A[j]$, then the pair $(A[i], A[j])$ is called an *inversion* of A.

   (a) List all inversions of the array $\{4, 2, 9, 1, 7\}$. **(5 points)**

   (b) What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many inversions does it have? **(5 points)**

   (c) Give a divide-and-conquer algorithm that computes the number of inversions in array A in $O(n\log n)$ time. (**Hint:** Modify merge sort.) **(20 points)**

4. **(20 points)** In SELECTION algorithm we studied in class, the input numbers are divided into groups of five and we choose the median of medians of these groups as the pivot. Now we partition A into groups of **three** and still choose their median of medians of groups, as the pivot instead. We can prove the algorithm has an upper bound $O(n)$ under this case. The analysis is as the following:

   We use p to denote the pivot, i.e., the median of medians of groups, and the time of computing the pivot is $T(n/3) + n$. In the partition step, we have ________ groups whose medians are smaller than p and thus in each such group, at least ________ numbers are smaller than p. Therefore, the size of A_1 is no less than ________, which leads the size of A_2 is no more than ________. Likewise, we have all left groups whose medians are larger than p and in each such group, there are at least ________ numbers larger than p. The size of A_2 is thus no less than ________, which leads the size of A_1 is no more than ________. The max size of A_1 and A_2 is no more than ________, denoted by X. Therefore, we obtain:

$$T(n) <= T(X) + T(n/3) + n$$

Now we prove $T(n) = O(n \log n)$ by using the substitution method to solving the above recurrence.

Guess: We guess $T(n) = O(n \log n)$. In other words, we want to find a constant c and n_0, such that $T(n) \leq c \cdot n \log n$ for all $n > n_0$.

Verification: We assume that the above is true for $T(X)$ and $T(n/3)$. Then, we can obtain the following:

$$T(n) <= \underline{\hspace{3em}} + n <= cn \log n$$

Our goal is to find a constant c and n_0, such that for all $n > n_0$, $T(n) \leq cn \log n$. If $c =$ ________ and $n_0 = 1$, then clearly it holds for all $n > n_0$. We conclude that our guess $T(n) = O(n \log n)$ is true. Therefore, if we do groups of three and apply the above way to select pivots, the time complexity of the SELECTION algorithm is $O(n \log n)$.

5. **(30 points)** Given a set A of n positive integers $\{a_1, a_2, \ldots, a_n\}$ and another positive integer M, find a subset of numbers of A whose sum is *closest* to M. In other words, find a subset A' of A such that the absolute value $|M - \sum_{a \in A'} a|$ is minimized, where $\sum_{a \in A'} a$ is the total sum of the numbers of A'. For the sake of simplicity, you only need to return the sum of the elements of the solution subset A' without reporting the actual subset A'.

   For example, suppose $A = \{1, 4, 7, 12\}$ and $M = 15$. Then, the solution subset is $A' = \{4, 12\}$, and thus your algorithm only needs to return $4 + 12 = 16$ as the answer.

   Let K be the sum of all numbers of A. Design a dynamic programming algorithm for the problem and your algorithm should run in $O(nK)$ time (note that it is not $O(nM)$).