COL351: Analysis and Design of Algorithms **Fall 2024**

Tutorial Sheet 1

Announced on: July 25 (Thurs)

Problems marked with ($\star$) will not be asked in the tutorial quiz.

1. Prove that $\log_2(n!) = \Theta(n \log_2 n)$.

2. Prove or disprove:
   a) $\log n = \mathcal{O}(n^\varepsilon)$ for any constant $\varepsilon > 0$.
   b) $4^n = \mathcal{O}(2^n)$.
   c) $n^2 + \mathcal{O}(n) = \mathcal{O}(n^2)$.
   d) $\lceil n \rceil = \Theta(n)$.

3. Identify and discuss the error(s) in the following false statement and its false proof. Mention the erroneous step(s) clearly.

   Claim: Let $f(n) = \sum_{i=1}^{n} i$. Then, $f(n) = \mathcal{O}(n)$.

   Proof: By induction on n.

   Let the induction hypothesis be $P(n) : f(n) = \mathcal{O}(n)$.

   Base case: $f(1)$ equals 1, which is $\mathcal{O}(1)$. Thus, $P(1)$ is TRUE.

   Induction step: Observe that $f(n+1) = f(n) + (n+1)$. Since $f(n) = \mathcal{O}(n)$ by induction assumption, and since $n + 1 = \mathcal{O}(n)$, we have that $f(n+1) = \mathcal{O}(n)$, which, in turn, is $\mathcal{O}(n+1)$. This proves the claim. □

4. Consider the following modification to the `MergeSort` algorithm: Divide the input array into *thirds* (rather than halves), recursively sort each third, and combine the results using a three-way `Merge` subroutine. What is the number of pairwise comparisons made by this algorithm as a function of the length n of the input array, ignoring constant factors and lower-order terms?

   (For the purpose of the tutorial quiz, you don't need to write the algorithm or its correctness analysis. Just highlight the key steps in your argument.)

5. Suppose the running time $T(n)$ of an algorithm is bounded by the recurrence with $T(1) = 1$ and

$$T(n) \leqslant T(\lfloor \sqrt{n} \rfloor) + 1 \text{ for } n > 1,$$

where $\lfloor x \rfloor$ denotes the *floor* function that rounds its argument down to the nearest integer. Provide the smallest correct upper bound on the asymptotic running time of the algorithm and justify your answer. Note that the master theorem does not apply.

6. Imagine you are a trader in a market in which a particular commodity can be bought only once and sold only once during a season. Naturally, the purchase much precede the sale. The price of the commodity fluctuates every day. The data from the last season is now available, and you want to know the maximum profit you could have made. Design an $\mathcal{O}(n)$ algorithm for this problem, where the input is the list of prices $\{p_1, p_2, \ldots, p_n\}$ for the n days in the last season and the output is the maximum possible profit. Assume that addition, subtraction, pairwise comparison, etc. takes unit time.

   Sample input prices: $70, 100, 140, 40, 60, 90, 120, 30, 60$.
   Output: Max profit: 80.

7. ($\star$) You are given as input an unsorted array of n distinct numbers, where n is a power of 2. Give an algorithm that identifies the second-largest number in the array while using at most $n + \log_2 n - 2$ pairwise comparisons.

8. ($\star$) Consider a tournament with n teams in which each team plays every other team exactly once, and each game has one winner (i.e., no ties). Unfortunately, the tournament is being held behind closed doors and you are unable to attend the games. You can, however, make a call to the organizers and, in each call, ask for the result of exactly one game. Say you want to find out if there is an *undisputed* champion, which is a team that wins all its games. Show that you can determine the existence of such a team with at most $2n - \lfloor \log_2 n \rfloor - 2$ calls.