Assignment 3: CS21003 Algorithms 1

Prof. Partha Pratim Chakrabarti and Palash Dey
Indian Institute of Technology, Kharagpur

Submit by 11:59 PM of March 22

1. Design an efficient data structure where the key elements are lower-case alphabetic strings and the ordering is lexicographic. Each string can be of maximum size m. The operations are insert, delete, find, max, and min. Clearly explain how you will

   (a) implement this data structure,

   (b) provide an example representation,

   (c) present algorithms for each operations,

   (d) analyze the time complexity of each operation,

   (e) space complexity for storing n elements to efficiently manage the structure.

   Do NOT assume that m is a constant.

   [10 Marks]

2. Suppose you are given a black box access to a sorting algorithm which takes an array containing distinct integers as input and outputs the sorted array. Use this algorithm in a black box fashion to design an algorithm to remove all duplicates from an integer array. You are allowed to perform $\mathcal{O}(n)$ comparisons other than the comparisons done inside the black box of the sorting algorithm. You are free to perform all other kind of computations of any time complexity.

   [10 Marks]

   Solution sketch. Let $\mathcal{A}[1, \ldots, n]$ be the input array. We define $\varepsilon = \frac{1}{2n} \min_{1 \leqslant i < j \leqslant n} |\mathcal{A}[i] - \mathcal{A}[j]|$. We define another array $\mathcal{B}[1, \ldots, n]$ as $\mathcal{B}[i] = \mathcal{A}[i] + i \times \varepsilon$. We now sort $\mathcal{B}$, from the sorted order of $\mathcal{B}$, we "find" (how?) the sorted order of $\mathcal{A}$, and remove duplicates. □

3. Suppose you are given a gray box access to a comparison-based algorithm for removing all duplicates from an integer array — you can observe the sequence of comparisons that the algorithm performs on any input and nothing else. Use this gray box to design a comparison-based algorithm to sort an integer array. You are allowed to perform $\mathcal{O}(n)$ comparisons other than the comparisons made by the gray box algorithm. You are free to perform all other kind of computations of any time complexity. Can you now see that $\Omega(n \log n)$ is also a lower bound on the number of comparisons that any comparison-based algorithm must perform to remove all duplicates? (You do not need to write in your answer script that yes I see! The last sentence is for your own understanding.)

   [10 Marks]

   Solution sketch. Let $\mathcal{A}[1, \ldots, n]$ be the set of integers that we need to sort. We execute our algorithm for removing duplicates on $\mathcal{A}$. Let $\mathcal{C}$ be the set of pairs of elements of $\mathcal{A}$ that are compared by our duplicate elimination algorithm. We thus know the order of every pair of elements in $\mathcal{C}$. We claim that, for every $1 \leqslant i < j \leqslant n$, we know the order between $\mathcal{A}[i]$ and $\mathcal{A}[j]$ from the comparisons of $\mathcal{C}$ by transitivity. If not, then there would exist two indices $1 \leqslant i < j \leqslant n$ such that the order between $\mathcal{A}[i]$ and $\mathcal{A}[j]$ does not follow from the orders of $\mathcal{C}$.

Suppose the duplicate elimination algorithm detects $\mathcal{A}[i]$ and $\mathcal{A}[j]$ to be the same (or not same respectively). We can then construct an array $\mathcal{A}$ which is consistent with the orders of $\mathcal{C}$ but $\mathcal{A}[i]$ and $\mathcal{A}[j]$ are not the same (or same respectively). This contradicts the correctness of the duplicate elimination algorithm. □