**cs473: Algorithms** Assigned: *Tue., Sep. 3, 2019*

# Problem Set #1

Prof. Michael A. Forbes
Prof. Chandra Chekuri

Due: *Wed., Sep. 11, 2019 (10:00am)*

Some reminders about logistics.

- **Submission Policy:** See the course webpage for how to submit your pset via gradescope.
- **Collaboration Policy:** For this problem set you are allowed to work in groups of up to three. Only one copy should be submitted per group on gradescope. See the course webpage for more details.
- **Late Policy:** Late psets are not accepted. Instead, we will drop several of your lowest pset problem scores; see the course webpage for more details.
- This problem set has two pages.

All (non-optional) problems are of equal value.

1. Given a sequence $a_1, a_2, \ldots, a_n$ of $n$ distinct numbers, an *inversion* is a pair $i < j$ such that $a_i > a_j$. Note that a sequence has no inversions if and only if it is sorted in ascending order. The book by Kleinberg-Tardos describes (see Chapter 5) an $O(n \log n)$ algorithm to count the number of inversions in a given sequence (*hint:* look at this algorithm). We consider two generalizations.

   (a) Call a pair $i < j$ a *valuable* inversion if $a_i > \alpha a_j$ where $\alpha \geq 1$ is a fixed constant. Describe an $O(n \log n)$ time algorithm to count the number of valuable inversions in a given sequence.

   (b) Consider a further generalization. In addition to the sequence $a_1, \ldots, a_n$ we are given weights $w_1, \ldots, w_n$ where $w_i \geq 1$ for each $i$. Now call a pair $i < j$ a significant inversion if $a_i > w_j a_j$. Describe an $O(n \log^2 n)$ time algorithm (or better!) to count the number of significant inversions given the sequences $a$ and $w$.

   Assume the sequences are given in arrays of size $n$.

2. Describe an algorithm to count the number of distinct 4-cliques in a given graph on $n$ nodes in time faster than $O(n^4)$ time. Recall that a *$k$-clique* is a complete graph on $k$ nodes. You can use the triangle counting algorithm discussed in lecture as a black box.

3. Describe an efficient algorithm that given two strings $A = a_1 a_2 \cdots a_n$ and $B = b_1 b_2 \cdots b_m$ outputs the smallest integer $k$ such that $A$ can be written as the concatenation of $k$ strings $A_1 \circ A_2 \circ \cdots \circ A_k$ (here '$\circ$' stands for string concatenation) and likewise $B$ can be written as $B_1 \circ B_2 \circ \cdots \circ B_k$ with the condition that the string $A_i \circ B_i$ is a palindrome for $1 \leq i \leq k$.

   For example, if $A = abac$ and $B = caba$ then $k = 1$ is possible by setting $A_1 = A$ and $B_1 = B$ since $A_1 \circ B_1 = A \circ B$ is a palindrome. Another example is $A = abc$ and $B = d$ in which case $k = 4$ is achieved by writing $A = a \circ b \circ c \circ \varepsilon$ and $B = \varepsilon \circ \varepsilon \circ \varepsilon \circ d$. Here $\varepsilon$ is the empty string which satisfies the property that for any string $C$, $C \circ \varepsilon = \varepsilon \circ C = C$.

4. (**optional**, *not* for submission) Let $A, B, C$ be three sorted arrays of integers with a total of $N$ elements. Assume for simplicity that all integers in the arrays are distinct. Describe an algorithm that given $A, B, C$ and an integer $k$ returns the $k$-th smallest element in $O(\log N)$ time. You may want to solve the case with two sorted arrays first. *Hint:* How can one discard a constant fraction of the elements in $O(1)$ time?