CS 473, Fall 2017
Homework 1 (due September 13 Wednesday at 8pm)

You may work in a group of at most 3 students. Carefully read `http://engr.course.illinois.edu/cs473/policies.html` and `http://engr.course.illinois.edu/cs473/integrity.html`. One member of each group should submit via Gradescope.

1. [*15 pts*] Given numbers $a_1, \ldots, a_n$, describe an efficient algorithm to compute the coefficients of the polynomial $(x-a_1)(x-a_2)\cdots(x-a_n)$. You may assume that each arithmetic operation takes constant time (don't worry about the sizes of the coefficients getting big).

   [Note: I believe getting exactly $O(n \log n)$ time is still open, but you should be able to get close (within logarithmic factors) to $O(n \log n)$.]

2. [*23 pts*] Given three sets of integers A, B, and C with a total of $|A|+|B|+|C| = n$ elements, we want to decide whether there exist elements $a \in A$, $b \in B$, and $c \in C$ such that $c = a + b$. It is not difficult to obtain a near $O(n^2)$ time algorithm for this problem. Here, we will explore subquadratic algorithms for special cases.

   (a) [*9 pts*] For the case when $A, B, C \subset \{1, \ldots, n^{1.9}\}$, give an algorithm that runs in $O(n^{1.9} \log n)$ time, by using FFT.

   (b) [*5 pts*] For the case when B and C have at most r elements (with no restriction on A), give an algorithm that runs in $O(r^2 \log n)$ time, assuming that A is stored in a *sorted* array.

   (c) [*9 pts*] For the case when $A \subset \{1, \ldots, n^{1.9}\}$ and B and C consist of integers but with no restrictions on the range, give an algorithm that runs in $O(n^t)$ time for some constant t strictly smaller than 2.

   [Hint: for B and C, divide into intervals of length $n^{1.9}$. For intervals with fewer than r elements of B and C, use one algorithm. For intervals with more than r elements of B and C (how many such intervals can there be?), use another algorithm. How should we set the parameter r?]

3. [*12 pts*] We are given a weighted complete graph $G = (V, E)$, where each pair of vertices u and v define a distance $d(u, v)$. Here, distances are symmetric (i.e., $d(u, v) = d(v, u)$). In the *2-center* problem, we want two vertices $u, v \in V$ such that

$$r_{uv} \ := \ \max_{w \in V} \min\{d(u, w), d(v, w)\}$$

   is as small as possible. (Motivation: we want to build two "hospitals" u and v, such that the maximum "response time" to any site is as small as possible.)

   Give a subcubic algorithm to solve this problem.

   [Hint: first solve the decision problem (given a value R, decide whether the optimal value $\min_{u,v \in V} r_{uv}$ is less than R).]