Problem Set 7

This problem set contains two theory questions. The problem set is due **Friday, May 5th** at **11:59PM**.
Solutions should be turned in through the course website. Your solution should be in PDF format using LaTeX. Remember, your goal is to communicate. Full credit will be given only to a correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Problem 7-1. [50 points] **Ghostbusters and Ghosts**

A group of n Ghostbusters is battling n ghosts. Each Ghostbuster carries a proton pack, which shoots a stream at a ghost, eradicating it. A stream goes in a straight line and terminates when it hits the ghost. The Ghostbusters decide upon the following strategy. They will pair off with the ghosts, forming n Ghostbuster-ghost pairs, and then simultaneously each Ghostbuster will shoot a stream at his chosen ghost. As we all know, it is *very* dangerous to let streams cross, and so the Ghostbusters must choose pairings for which no streams will cross.

Assume that the position of each Ghostbuster and each ghost is a fixed point in the plane and that no three positions are collinear.

(a) [25 points] Argue that there exists a line passing through one Ghostbuster and one ghost such that the number of Ghostbusters on one side of the line equals the number of ghosts on the same side. Describe how to find such a line in $O(n \lg n)$ time.

Solution: Find the bottom, left-most point as in Graham scan. Sort the remaining points (by angle) from that point. Assume that the bottom, left most point is a Ghostbuster. Visit the sorted points by increasing angle, keeping track of the difference between number of visited Ghostbusters and Ghosts. Stop when the difference is -1, and connect the point to the bottom, left-most point. Run time is dominated by the sort, which takes $O(n \lg n)$-time.

(b) [25 points] Give an $O(n^2 \lg n)$-time algorithm to pair Ghostbusters with ghosts in such a way that no streams cross.

Solution: Using the above algorithm matches one pair of Ghostbuster and Ghost. On each side of the line formed by the pairing, the number of Ghostbusters and Ghosts are the same, so use the algorithm recursively on each side of the line to find pairings. The worst case is when, after each iteration, one side of the line contains no Ghostbusters or Ghosts. Then, we need $n/2$ total iterations to find pairings, giving us an $P(n^2 \lg n)$-time algorithm.

Problem 7-2. [50 points] **Three is company**

With MIT set to increase the size of the incoming class, the evil Housing Office has decided to turn some East Campus doubles into triples. The existing residents of those rooms are understandably concerned about getting saddled with somebody lame, and so they put their considerable overengineering prowess to work on the problem.

On the way back from putting an ironic protest installation on top of the dome, one of the students responsible for making housing arrangements has an epiphany (or, at the least, heavily caffeinated inspiration) and proposes the following scheme. Say that Alice and Bob are the existing roommates; to determine their compatibility with a prospective freshman, they each choose a set of n distinct integers in the range $\{0, \ldots, m\}$ (A and B, respectively) which correspond to their responses to a survey.

Each freshman will also be asked the same questions, producing a similar set C. Alice and Bob will be considered compatible with that freshman if there is some $a \in A$, $b \in B$, and $c \in C$ such that $a + b = c$. Describe an $O(m^{\log_2 3})$-time algorithm for determining whether the prospective roommate is compatible with Alice and Bob. (*Hint:* represent each set as a sequence of 0-1 coefficients of a polynomial.)

Solution: Create polynomials $\hat{A}(x)$ and $\hat{B}(x)$ of degree m such that $a_i = 1$ iff $i \in A$ and $b_j = 1$ iff $j \in B$. Compute $\hat{D}(x) = \hat{A}(x) \cdot \hat{B}(x)$. Note that $d_k > 0$ iff there are $i \in A$ and $j \in B$ such that $k = i + j$ and $a_i = 1$ and $b_j = 1$. Now we just need to check whether $d_k = c_k = 1$ for any k.

Each polynomial can be created in $O(m)$ time. We use Karatsuba's algorithm for multiplying the polynomials, which takes $O(m^{\lg_2 3})$ time. Checking to see if $d_k = c_k = 1$ takes linear time. Overall, the running time is $O(m^{\lg_2 3})$.