CSE 202 Homework 2
Greedy algorithms and Divide-and-Conquer
Due Tuesday, May 20

For each of the algorithm problems, design as asymptotically efficient an algorithm as possible. Give a correctness argument (explanation, if it is relatively simple, or proof if not) and time analysis. You may use any well-known algorithm or data structure, or algorithm from the text or from class, as a sub-routine without needing to provide details.

**Shifts** Problem 15 on page 196 of the textbook.

**Oxen pairing** Give an efficient algorithm for the following problem: We have $n$ oxen, $Ox_1, ..Ox_n$, each with a strength rating $S_i$. We need to pair the oxen up into teams to pull a plow; if $Ox_i$ and $Ox_j$ are in a team, we must have $S_i + S_j \geq P$, where $P$ is the weight of a plow. Each ox can only be in at most one team. Each team has exactly two oxen. We want to maximize the number of teams. [5 points correct algorithm, 12 points correctness proof, 3 points efficiency]

**Approximate min cost bisection clustering** A $[1-2]$ metric on $2n$ points is a symetric distance function $1 \leq d(x,y) = d(y,x) \leq 2$ for each points $x \neq y$. ($d(x,x) = 0$ by convention.) (Note that such a function is always a metric, i.e., always obeys the triangle inequality $d(x,z) \leq d(x,y) + d(y,z)$.) A bisection clustering of the points divides them into two equal size disjoint sets $S$ and $T$ with $|S| = |T| = n$. You want to find a bisection clustering that minimizes the sum of edge weights within a cluster, $\sum_{x,y \in S} d(x,y) + \sum_{x,y \in T} d(x,y)$. Find the best polynomial time approximation algorithm you can for this problem. (Most points based on approximation ratio. Ratio 1.5 is worth most of the points.)

**Base Conversion** . Give an algorithm that inputs an array of $n$ base base 10 digits representing a positive integer and outputs an array of bits representing the same integer in base 2. Your algorithm should be $o(n^2)$, strictly better than the time asked for on the calibration homework. You will probably need to use a divide-and-conquer strategy, and use a fast integer multiplication sub-routine (from class). [3 points correct algorithm and correctness proof, 7 points efficiency]

**Implementation: Integer Multiplication** Implement the $O(n^{\log 3})$ divide-and-conquer algorithm for integer multiplication from class, but with a threshold, below which naive "gradeschool" multiplication is used. Use an array of digits to represent inputs and outputs. Experimentally determine the optimal threshold. For what values of $n$ do you see an improvement in the time using divide-and-conquer, both using no threshold and using the optimal threshold?

When describing an algorithm, don't write out an entire pseudo-code; just describe it at a high level. Be sure to specify completely all data structures used in the algorithm. Include correctness proofs and time analysis for all algorithms, except for the implementation problem.