Problem Set 2: Divide And Conquer and Dynamic Programming

Due: Friday 10/15 11:59PM

Feel free to work in groups of at most 4 for these - if you have a group of more than 4, please run it by me first. If you do work in a group, please include the names of those that you worked with. **However: each student should submit a separate copy where the solutions have been written by yourself.**

1. (10 points) Recall the inversion problem we worked through in class. Given a list of n items $(e_1, ..., e_n)$, we say (e_i, e_j) is a mega-inversion if $e_i > 3e_j$ and $i < j$. Design a $O(n \log n)$-runtime algorithm that calculates the number of mega-inversions and convince me of its correctness.

2. (10 points) Imagine your buddy and you are playing a **very fun** version of 20 questions: your bud has two separate lists (say list A and list B), each of n values written down behind their back. No value is repeated (meaning if 3 is in A then 3 is not in B). You are allowed to specify a list and ask for the kth smallest item, to which they must truthfully respond. So for example, you could ask for the 3rd smallest item of list B, etc. Determine a strategy of questions that will determine the median (we'll say, the nth smallest item) of the two lists combined in $O(\log n)$ questions.

3. (10 points) Suppose you are given $T(n) = T\left(\frac{n}{2}\right) + O(n)$. Using the tree based method described in class, show that $T(n) = O(n)$.

4. The input of this problem is a list of positive numbers $(v_1, ..., v_n)$. The optimal solution is a collection of the numbers such that no two sequential numbers are taken (so if v_i is taken then v_{i-1} and v_{i+1} are off limits) but the summation of the collection is maximal.

   (a) (5 points) Consider the following 'greedy' algorithm: sequentially choose the greatest numbers not yet chosen that are not directly before or after any numbers already chosen. Show that this algorithm is incorrect.

   (b) (5 points) Consider the following algorithm: let C_1 be the collection of all even indexed numbers, and C_2 the collection of all odd indexed numbers, and select the collection which has greater total summation. Show that this algorithm is incorrect.

   (c) (10 points) Design a dynamic programming solution and convince me that it correct and has polynomial runtime.