Recall that your solution must be typed via Latex. Also, remember that this is a group assignment; group size must be 2. Since our class has an odd number of students, I allow exactly one group to have size 3. If you want to form a group of size 3, email me to make sure that there’s one such group.

**Deliverables:** Submit to Canvas two PDF files: a writeup for the homework solution, and a transcript of the discussion of your group members. Both files should contain the names of all group members.

1. (70 points) In class, we studied the problem of counting the number of inversions in an array $A[1 : n]$. We motivated this problem as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let’s call a pair $(i, j)$ a **significant inversion** if $i < j$ and $A[i] > 2A[j]$. Given an $O(n \log(n))$ algorithm to count the number of significant inversions in an array $A[1 : n]$.

   [We expect that your algorithm is written in pseudocode, together with an English description and a short, informal argument for its correctness and running time.]

2. (70 points) You are given a binary string $x$ and an array $A[1 : n]$ of binary strings. Assume that $x$ and the elements of $A$ have the same length. Let $\oplus$ denote the xor operator on binary strings. For example $1010 \oplus 1101 = 0111$, and $1110 \oplus 1111 = 0001$. Assume that xor’ing two strings takes $O(1)$ time. Give a divide-and-conquer algorithm to check if there’s a subarray $A[i : j]$ of $A$ such that $A[i] \oplus \cdots \oplus A[j] = x$. Your algorithm should return such a pair $(i, j)$ if they exist; otherwise return ($-1, -1$). Your algorithm should run in $O(n \log(n))$ time.

   [We expect that your algorithm is written in pseudocode, together with an English description and a short, informal argument for its correctness and running time.]

   **Note:** If you are not familiar with the xor operator, try to think of it as the addition operator; the two operators have many similar properties. Still, the xor operator has this nice property: if you have $x = a \oplus b$ then $x \oplus a = b$.

3. (60 points) a) (30 points) Let

   $$T(n) = \begin{cases} 
   1 & \text{ if } n = 1 \\
   4T(n/2) + n^2 \log_2(n) & \text{ otherwise}
   \end{cases}$$

   Use the recursion tree method, show that $T(n) = O(n^2 \log^2(n))$. You can assume that $n$ is a power of 2.

   [We expect the drawing of the recursion tree to derive a summation, and a rigorous justification of the upper bound of the sum.]

   b) (30 points) Let

   $$T(n) = \begin{cases} 
   1 & \text{ if } n < 8 \\
   T([n/2]) + T([n/4]) + T([n/8]) + n & \text{ otherwise}
   \end{cases}$$

   Use the substitution method, obtain a Big-Theta bound for $T(n)$. 

2-1