Problem 1 (Faster Algorithm for Finding the Number of Crossing Inversions). Let $S_1$ and $S_2$ be two disjoint sets of $n$ integers. Assume that $S_1$ is stored in an array $A_1$, and $S_2$ in an array $A_2$. Both $A_1$ and $A_2$ are sorted in ascending order. Design an algorithm to find the number of such pairs $(a, b)$ satisfying all of the following conditions: (i) $a \in S_1$, (ii) $b \in S_2$, and (iii) $a > b$. Your algorithm must finish in $O(n)$ time (we gave an $O(n \log n)$-time algorithm in the class).

Problem 2. Give an $O(n \log n)$-time algorithm to solve the dominance counting problem discussed in the class. (Hint: Using the result of Problem 1)


1. Give an algorithm to find a sub-array of with the largest weight, among all sub-arrays $A[i : j]$ with $j = n$. Your algorithm must finish in $O(n)$ time.

2. Give an algorithm to find a sub-array with the largest weight in $O(n \log n)$ time (among all the possible sub-arrays).

Problem 4. In the class, we explained how to multiply two $n \times n$ matrices in $O(n^{2.81})$ time when $n$ is a power of 2. Explain how to ensure the running time for any value of $n$. 