Chapter 5

1. (7) Problem 1 This problem can be solved using divide and conquer. A and B are the two databases and $i^{th}$ is the smallest element of A and B. First let's compare the medians of the two databases. Let $k$ be $\frac{1}{2}n$ then $A(k)$ and $B(k)$ are two medians. If $A(k) < B(k)$ then $B(k)$ is always greater than first $k-1$ elements of B, it is at least $2^k$th element in the combined database. Since $2k \geq n$, all elements are greater than $B(k)$ are greater than median, so we can eliminate the second half of $B$. This will help you decide which part of the array to look next and which to eliminate, you will then apply the same recursive strategy to the part which is left. The recurrence for the running time is then

$$T(n) = T(n/2) + 2 = O(\log n)$$

The running time - the solution can be found using the recurrence tree.

2. (5) Solve the recurrence $T(n) = 4T(n/2) + n$ appealing to recursion tree
3. (5) Solve the recurrence \( T(n) = T(n/2) + n^2 \) by a method of your choice

\[ T(n) = T(n/2) + n^2 \]

\[
\begin{cases}
T(n) & \text{if } \log_2 n = 0 \\
T(n/2) & \text{if } \log_2 n = 1 \\
\vdots & \\
T(1) & \text{if } \log_2 n \rightarrow \infty \\
\end{cases}
\]

We know:

\[
\sum_{n=0}^{\infty} x^n = \frac{1}{1-x} \quad \leq C_0 n^2
\]

Hence above summation can be bound from above by the following constant:

\[
\sum_{n=0}^{\infty} \left( \frac{1}{2n} \right)^2 = \frac{1}{1-\frac{1}{2}}
\]

4. (7) Problem 2: We will design a recursive divide and conquer algorithm for counting significant inversions. The main difference will be that in the merge stage we merge twice: first merge \( b_1, \ldots, b_k \) with \( b_{k+1}, \ldots, b_n \) just for sorting and then we merge \( b_1, \ldots, b_k \) with \( 2b_{k+1}, \ldots, 2b_n \) for counting significant inversions. In the merge step the ALG returns \( N_1 \) and \( b_1, \ldots, b_k \) and \( N_2 \) and \( b_{k+1}, \ldots, b_n \) which are sorted and \( N_1 \) and \( N_2 \) are the numbers of significant inversions. Then we need to compute the number of significant inversion \( N_3 \) and returns \( N_1 + N_2 + N_3 \).

We will need to implement a variation of merge-count, as we initialize \( N_3 \) to 0 and as we are merging we check:

- if \( b_i \leq 2b_j \) then
  - if \( j > k + 1 \) decrease \( j \) by 1
  - if \( j = k + 1 \) return \( N_3 \)

- if \( b_i > 2b_j \) then increase \( N_3 \) by \( j - k \)
  - if \( i > 1 \) decrease \( i \) by 1
  - if \( i = 1 \) return \( N_3 \)