Problem 1. Consider that we run the divide-and-conquer algorithm to solve the dominance screening on the following 2D dataset (i.e., report the white points that are not dominated by the black ones). The algorithm divides the dataset along the x-dimension in halves, where the first half consists of points $p_1, p_2, p_3, p_4, p_5$, and the second consists of $p_6, p_7, p_8, p_9, p_{10}$. Answer the following questions:

- What is the answer set returned from each half?
- In order to merge the two answer sets into the final answer, the algorithm needs to solve a 1D dominance screening instance. Indicate the 1D points (by specifying their coordinates) in this instance.

Solution. \{p_2\} is returned from the first half, and \{p_7, p_8\} is returned from the second half.

The 1D dominance screening instance is to report all the white points that are not dominated by any black point, where (i) the 1D black points are 9, 6 and 4, i.e., the y-coordinates of points $p_1, p_3$ and $p_5$, respectively; (ii) the 1D white points are 5 and 2, i.e., the y-coordinates of points $p_7$ and $p_8$.

Problem 2. Consider that we run the divide-and-conquer algorithm to find the skyline on the following dataset (i.e., report all the points that are not dominated by any other point). The algorithm divides the dataset along the x-dimension in halves, where the first half consists of points $p_1, p_2, p_3, p_4, p_5$, and the second consists of $p_6, p_7, p_8, p_9, p_{10}$. Answer the following questions:

- What is the answer set returned from each half?
- In order to merge the two answer sets into the final answer, the algorithm needs to solve a 1D dominance screening instance. Indicate the 1D points (by specifying their coordinates) in this instance.
Solution. \( \{p_1, p_2\} \) is returned from the first half, and \( \{p_6, p_8\} \) is returned from the second half.

The 1D dominance screening instance is to report all the white points that are not dominated by any black point, where (i) the 1D black points are 9 and 4, i.e., the \( y \)-coordinates of points \( p_1 \) and \( p_2 \); (ii) the 1D white points are 7 and 2, i.e., the \( y \)-coordinates of points \( p_6 \) and \( p_8 \).

Problem 3. Consider that we run the divide-and-conquer algorithm to find the skyline on the following 3D dataset:

\[
\begin{align*}
p_1 &= (1, 2, 3), p_2 = (2, 4, 6), p_3 = (4, 7, 9), p_4 = (4, 9, 3), p_5 = (6, 3, 1), \\
p_6 &= (7, 2, 9), p_7 = (8, 4, 5), p_8 = (8, 6, 7), p_9 = (9, 9, 8), p_{10} = (9, 2, 2).
\end{align*}
\]

The algorithm divides the dataset along the x-dimension in halves, where the first half consists of points \( p_1, p_2, p_3, p_4, p_5 \), and the second consists of \( p_6, p_7, p_8, p_9, p_{10} \). Answer the following questions:

- What is the answer set returned from each half?
- In order to merge the two answer sets into the final answer, the algorithm needs to solve a 2D dominance screening instance. Indicate the 2D points (by specifying their coordinates) in this instance.

Solution. \( \{p_1, p_3\} \) is returned from the first half, and \( \{p_6, p_7, p_{10}\} \) is returned from the second half.

The 2D dominance screening instance is to report all the white points that are not dominated by any black point, where (i) the 2D black points are \((2, 3)\) and \((3, 1)\), i.e., the projections on the \(yz\)-plane of points \( p_1 \) and \( p_5 \); (ii) the 2D white points are \((2, 9), (4, 5)\) and \((2, 2)\), i.e., the projections on the \(yz\)-plane of points \( p_6, p_7 \) and \( p_{10} \), respectively.

Problem 4. Consider the 2D dominance screening problem again. Recall that we have two sets \( P, Q \) of points in \( \mathbb{R}^2 \). The goal is to identify all the points \( q \in Q \) such that \( q \) is not dominated by any point in \( P \). Suppose that \( P \) and \( Q \) have already been sorted by x-dimension. Give an algorithm to solve the problem in \( O(n) \) time where \( n = |P| + |Q| \). You may assume that all the points in \( P \cup Q \) have distinct x-coordinates.

Solution. We scan the points of \( P \) and \( Q \) in ascending order of \( x \)-coordinate. In the meantime, maintain \( y_{\text{min}} \) as the smallest \( y \)-coordinate of the points from \( P \) seen so far; \( y_{\text{min}} = \infty \) at the beginning. We process the next point \( s \) as follows:

- If \( s \) comes from \( P \), update \( y_{\text{min}} \) if it is larger than the \( y \)-coordinate of \( s \).
• If \( s \) comes from \( Q \), report \( s \) if its \( y \)-coordinate is smaller than \( y_{\text{min}} \).

It is easy to verify that the algorithm runs in \( O(n) \) time.

**Problem 5*. Consider the 3D dominance screening (DS) problem. Recall that we have two sets \( P, Q \) of points in \( \mathbb{R}^3 \). The goal is to identify all the points \( q \in Q \) such that \( q \) is not dominated by any point in \( P \). Give an algorithm to solve the problem in \( O(n \log n) \) time where \( n = |P| + |Q| \). Again, you may assume that all the points in \( P \cup Q \) have distinct x-coordinates.

**Solution.** From \( P \), produce two sorted lists: (i) the \( x \)-list, where the points of \( P \) are sorted by \( x \)-coordinate, and (ii) the \( y \)-list, where the points are sorted by \( y \)-coordinate. In other words, each point of \( P \) is stored twice, once in each list. Do the same from \( Q \). This can be done in \( O(n \log n) \) time.

Our algorithm runs in almost the same way as described in the class, except that we divide the input sets in a more careful manner to preserve the point ordering:

• Divide the points of \( P \) and \( Q \) along the \( x \)-dimension in \( O(n) \) time as follows:
  - Pick the median \( x \)-coordinate value \( x_{\text{median}} \) among all the points in \( P \cup Q \).
  - Divide the points of \( P \) into \( P_1 \) and \( P_2 \), such that \( P_1 \) contains all the points with \( x \)-coordinates at most \( x_{\text{median}} \), and \( P_2 \) the rest. We need to produce an \( x \)-list and a \( y \)-list for \( P_1 \), and the same also for \( P_2 \). The \( x \)-lists of \( P_1 \) and \( P_2 \) can be obtained simply by splitting the \( x \)-list of \( P \) at the middle. To obtain their \( y \)-lists, we scan the \( y \)-list of \( P \); for each point \( p \) encountered, append it to the \( y \)-list of either \( P_1 \) or \( P_2 \), depending on the comparison of the \( x \)-coordinate of \( p \) and \( x_{\text{median}} \).
  - In the same manner, divide the points of \( Q \) into \( Q_1 \) and \( Q_2 \), and produce their \( x \)- and \( y \)-lists, respectively.

• Solve recursively the left DS instance with input sets \( P_1 \) and \( Q_1 \), and also the right instance with input sets \( P_2 \) and \( Q_2 \). Let \( A_{\text{left}} \) and \( A_{\text{right}} \) be their answer sets. We place the recursion requirement that both \( A_{\text{left}} \) and \( A_{\text{right}} \) be sorted by \( y \).

• Construct a 2D DS instance of \( P' \) and \( Q' \), where \( P' \) (\( Q' \), resp.) is obtained by projecting all the points of \( P_1 \) (\( Q_1 \), resp.) onto the \( yz \)-space. As all the points in \( P' \) and \( Q' \) are sorted by \( y \)-dimension, the algorithm in Problem 4 solves the problem in \( O(n) \) time.

• Combine \( A_{\text{left}} \) and \( A' \) into a set \( A \) sorted by \( y \) in order to fulfill the recursion requirement. As both \( A_{\text{left}} \) and \( A' \) are sorted by \( y \), this takes \( O(n) \) time.

• Return \( A \).

Let \( f(n) \) be the time of our algorithm on \( n \) points. It holds that \( f(n) \leq 2 \cdot f(n/2) + O(n) \) and \( f(1) = O(1) \). Thus, \( f(n) = O(n \log n) \).

**Problem 6.** Give an algorithm to find the skyline of \( n \) points in \( \mathbb{R}^d \) (where \( d \geq 3 \)) in \( O(n \log^{d-2} n) \) time.

**Solution.** Plugging in the algorithm in Problem 5 into the analysis discussed in the class, we know that the \( d \)-dimension dominance screening problem can be solved in \( O(n \log^{d-2} n) \) time for \( d \geq 3 \).

Then, plug in this result into the skyline analysis discussed in the class, we know that the \( d \)-dimension skyline problem can be solved in \( O(n \log^{d-2} n) \) time for \( d \geq 3 \).