Problem 1. Let R be a set of axis-parallel rectangles and P be a set of points, all in $\mathbb{R}^2$. We want to find report all the rectangle-point pairs $(r, p) \in R \times P$ such that r covers p. Design an algorithm to do so in $O(n \log n + k)$ time, where $n = |R| + |P|$ and k is the number of pairs reported.
(Hint: Both planesweep and divide-and-conquer will work).

Solution. **(Divide and Conquer)** For initialization, sort all objects by their y-coordinates, this is done only once, in $O(n \log n)$ time. We are then able to solve the 1D instance in $O(n)$ time. Build a hash table, and we can insert/delete an interval in $O(1)$ time. Reporting the point p included in these intervals will at most take $O(k)$ time.

All other steps follow what was taught:

1. Let R_{span} be the set of rectangles that do not define x-coordinates;
2. Solve the 1D instance on R' and P', where R' and P' are obtained by projecting R_{span} and P on the y-axis;
3. Remove R_{span} from R;
4. Divide the input x-coordinates equally with a vertical line ℓ, this is done by calculating the median of the x-coordinates in $O(n)$ time.
5. Preserve the order of sorted y-coordinates, let R_1 (or R_2) be the set of rectangles that intersect with the left (right) side of ℓ, P_1 (or P_2) be the set of points that fall on the left (right) side of ℓ;
6. Solve the left subproblem with input R_1, P_1 and right subproblem with input R_2, P_2, until either no points or no rectangles are included.

Solving the recurrence $f(n) = 2f(n/2) + O(n)$ will give $f(n) = O(n \log n)$.

(Planesweep) We sweep a vertical line ℓ from $x = -\infty$ to $x = +\infty$.

Sort all points, vertical boundaries of rectangles in $O(n \log n)$ time, which enables efficient event processing.

Then, we require a 1D data structure supporting:

- Inserting an interval in form of $[x_1, x_2], (-\infty, x_2], [x_1, +\infty)$;
- Deleting an interval existing;
- Given a point $p = x_0$, report all the intervals covering the point p.

A segment tree (to utilize this structure without error, we discretialize all coordinates before processing, then no undefined boundaries require care) could support the above operations satisfying the time constraints. Insertion and deletion should cost only $O(\log n)$ of time each, and reporting costs $O(\log n + k)$ time each.

Upon encountering a left vertical boundary of the rectangle $x \times [y_1, y_2]$, insert $[y_1, y_2]$ into the structure defined above; Upon sweeping through a point (x_0, y_0), we report the intervals covering this point (simply query with y_0); For the right boundary of the rectangle, remove the left boundary added beforehand.

Problem 2. Let R be a set of axis-parallel rectangles and P be a set of points, all in $\mathbb{R}^d$. We want to find report all the rectangle-point pairs $(r, p) \in R \times P$ such that r covers p. Design an algorithm to do so in $O(n \text{ polylog } n + k)$ time, where $n = |R| + |P|$ and k is the number of pairs reported.

Solution. **(Divide and Conquer)** We utilize the fact that the 2D-instance of the problem can be solved in $O(n \log n)$ time.

The only difference is that we now need to deal with the edge case before solving the subproblem (i.e. decrease the dimension by 1 with projection on the spliting hyperplane).

1. Let R_{span} be the set of rectangles that do not define the coordinates in the d-th dimension;
2. Solve the instance with dimension $d-1$ on R' and P', where R' and P' are obtained by projecting R_{span} and P on the hyperplane separates the positive half and the negative half of the d-th dimension;
3. Remove R_{span} from R;
4. Divide the input coordinates (in the d-th dimension) equally with a hyperplane α orthogonal to the d-th dimension.
5. Let R_1 (or R_2) be the set of rectangles that intersect with the left (right) side of α, P_1 (or P_2) be the set of points that fall on the left (right) side of α;
6. Solve the left subproblem with input R_1, P_1 and right subproblem with input R_2, P_2, until either no points or no rectangles are included.

Solving the recurrence $f(n, d) = 2f(n/2, d) + g(n, d-1)$ where $g(n, 2) = O(n \log n)$ will give $f(n) = O(n \log^{d-1} n)$ for $n \geq 2$.

Problem 3. Solve the dominance screening problem in 3D space in $O(n \log n)$ time. (Hint: Planesweep).

Solution. To apply the planesweep algorithm, we sort the points in P, Q, first by their z-coordinates, then the x-coordinates, finally by their y-coordinates, all in descending order.

The algorithm scans a plane from $z = -\infty$ to $z = +\infty$, handling a 2D instance one at a time. Note that the plane getting scanned afterwards has points with z-coordinates no larger than the ones lying on the previous planes. These points are dominated by some points in at least one axis.

We will need a data structure supports the following in $O(\log n)$ per operation, where data are saved in the form of (x, y):

1. Report $\max_{x_i \geq x_0} y_i$ for any (x_i, y_i) currently in the data structure, given x_0;
2. Insert a point (x_0, y_0).
3. No deletion required.

This can be implemented by a inverted binary indexed tree (fenwick tree), by discretializing all points before executing the algorithm.

Our algorithm works as follows:

```
procedure 3DMaxima(points in $P, Q$, $p_1, p_2, \ldots, p_n$ sorted as described)
    for points $p_{z,1}, p_{z,2}, \ldots, p_{z,t}$ on each $z$-coordinate do
        for every point $p_{z,i}$ do
            if $p_{z,i} \in Q$ and Query($x$ coordinate of $p_{z,i}$) $<$ $y$ coordinate of $p_{z,i}$ then
                output $p_{z,i}$
            end if
            if $p_{z,i} \in P$ then
                Insert($x$ coordinate of $p_{z,i}$, $y$ coordinate of $p_{z,i}$)
            end if
        end for
    end for
end procedure
```

The two **for** statements iterate through every points and thus run with at most $O(n)$ time.

The algorithm runs in $O(n \log n)$ time.

Problem 4. Let P be a set of n points in $\mathbb{R}^3$. Describe an algorithm to find all the maximal points of P in $O(n \log n)$ time.

Solution. We just need to tweak the algorithm for a little bit:

```
procedure 3DMaxima(points $p_1, p_2, \ldots, p_n$ sorted as described)
    for points $p_{z,1}, p_{z,2}, \ldots, p_{z,t}$ on each $z$-coordinate do
        for every point $p_{z,i}$ do
            if Query($x$ coordinate of $p_{z,i}$) $<$ $y$ coordinate of $p_{z,i}$ then
                output $p_{z,i}$
            end if
            Insert($x$ coordinate of $p_{z,i}$, $y$ coordinate of $p_{z,i}$)
        end for
    end for
end procedure
```

Problem 5. Let P be a set of n points in $\mathbb{R}^d$, where $d \geq 3$ is a constant. Describe an algorithm to find all the maximal points of P in $O(n \log^{d-2} n)$ time.

Solution. **(Divide and Conquer, Dimensionality Reduction)**

We solve the maximal point problem in 3D in $O(n \log n)$ time. This serves as a base case of the divide and conquer algorithm.

1. Divide the input x-coordinates equally with a vertical plane α;
2. Let P_1 (or P_2) be the set of points in P that lies on with the left (right) side of α, Q_1 (or Q_2) be the set of points in Q that fall on the left (right) side of α;
3. Solve the left subproblem with input P_1, Q_1 and right subproblem with input P_2, Q_2, until either no points or no rectangles are included;
4. Merging the subproblems: project all points reported by the left subproblem instance, and all points reported by the right subproblem instance to the hyperplane α;
5. Solve this projected problem: it should be have dimensionality of $d - 1$.

The above described algorithm's time complexity has the form $f(n, d) = 2f(n/2, d) + g(n, d - 1)$, where $g(n, 3) = O(n \log n)$. Solving the recursion gives $f(n) = O(n \log^{d-2} n)$.