(1)

Idea

Key observation is that the box is contained on the axis, so that for all points not bigger than its height, they generates a window (L, R) that the box must be in. (Note, the algorithm assume inside means exactly inside, so the point on the boundary is not considered as inside. But this should be a minor issue, since minor fix in algorithm will make it account for the other case).

Algorithm

- Let's make the vertical line $x = 0$. This requires shifting all points and boxes, this only takes $O(n)$ time. This makes the algorithm cleaner.
- Sort all boxes by their height h in nondecreasing order. Sort all points by their y in nondecreasing order.
- Initialize a window $[L, R]$ to be $[-\infty, \infty]$.
- Start from smallest boxes R with height R_h. Move through the points (in increasing y) P with $P_y < R_h$. Update the window for each such point P:
  - If $P_x < 0$, then the point is a constraint on the left, update $L = \max(L, P_x)$.
  - If $P_y \geq 0$, then the point is a constraint on the right, update $L = \min(R, P_x)$
- Now, if $R_l \geq L$ and $R_r \leq R$, then we find that box is empty.
- Otherwise check the next smallest box. If process terminates with no box remaining, that means no empty box exists.

Correctness

It should be clear that the box R is empty if and only if all points P such that $P_y < R_h$ must have $P_x \leq R_l$ and $P_y \geq R_r$. Since the box R intersects with line ℓ, that means $R_l \leq 0$ and $R_r \geq 0$. We could observe that the window $[L, R]$ makes sure that for all points, either $P_x < 0$ thus $P_x \leq L \leq R_l$ or $P_y \leq 0$ and $R_r \leq R \leq P_x$. So that makes sure the box R is indeed empty.

Time Complexity

Sorting points and and rectangles each takes $O(n \log n)$ time. Checking (and shifting) is one pass and takes $O(n)$ time. So the final complexity is $O(n \log n)$ time.

(2)

Idea

It should be straightforward that we could split the boxes by choose an axis, then for the boxes on the axis, call the subroutine, for the boxes on the left and right side, go to the sub cases.

Algorithm

- Sort all boxes in by their horizontal midpoints.
- Select the divide line ℓ at the median of all midpoints, so that roughly half the boxes are on the left, and half are on the right.
    - Call R_L the set of boxes such that $R_r < \ell$ (completely on left)
    - Call R_R the set of boxes such that $R_l > \ell$ (completely on right)
    - Call R_C the set of boxes intersects with ℓ
    - Call P_L the set of points such that $P_x < \ell$
    - Call P_R the set of points such that $P_x \geq \ell$
- Solve the subproblem on the R_L with the points P_L. Solve the subproblem on R_R with points P_R, call the subroutine in part (a) with R_C, P, and ℓ
- Reports empty boxes if either R_L, R_R or R_C reports an empty box. Return none otherwise.

Correctness

It should be obvious that it works, for subproblem we use only part of the points, because other parts of the point won't intersect with the boxes in the first place.

Time Complexity

Let $T(n)$ denotes the time to solve the problem, we see that the subproblem on the left and right is roughly half of the size $n/2$ as the divide line ℓ is at the median of all midpoints. The subroutine takes $O(n \log n)$ at worst case when all $R_C = R$. So:

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

By master theorem, so

$$T(n) = O(n \log^2 n)$$