P5

Monday, February 2, 2026 8:02 PM

b)

The list is sorted, so start from the end, at the greatest x-coordinate. $O(1)$

Add this coordinate to the list, staircase and save its y-value as stair_height. $O(1)$

Iterating from greatest to least, if an element in the list has a $y >$ stair_height, add the coordinate to staircase and stair_height := y. $O(n)$

Repeat until reaching end of list.

c)

Find the rightmost point, $O(n)$

and add it to list "staircase". $O(1)$

Let current ← rightmost point $O(1)$

From current point, search all points. If any point lies below this point, ignore it. From these points, choose the point closest in x-distance.

Add this point to staircase.

Current $\leftarrow$ closest in x-distance

Repeat until no point lie above current

$O(n)$

$O(n + 1 + 1 + n(h+1))$

$h+1$, because an additional check to ensure no point lie above the last point.

$O(n + n + nh)$

$O(nh)$

a)

Divide & Conquer

- We split all points until they are in groups of 1.

There are 4 cases for any two points, say a and b:

1 $a_x > b_x$ and $a_y > b_y$

2 $a_x = b_x$ and $a_y > b_y$

2 $a_x = b_x$ [illegible]

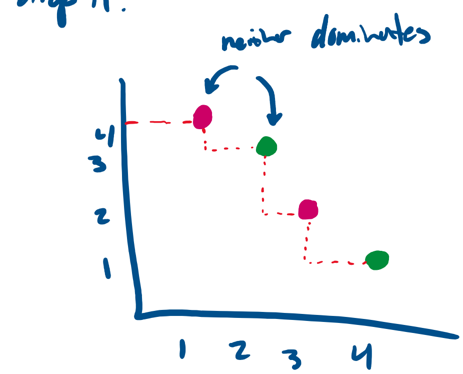
3 $a_x > b_x$ and $a_y = b_y$

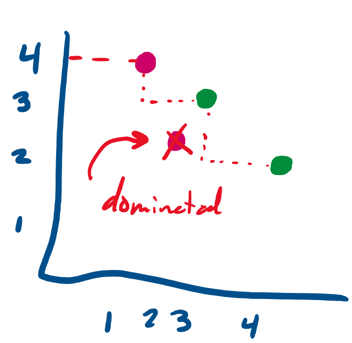
4 $a_x > b_x$ and $a_y < b_y$

In cases 1, a 'dominates' b, meaning we can simply drop the point because it cannot be part of the staircase. a passes on.

In case 2,3,4, neither dominate and both can be part of the staircase, sort by x and pass a and b on.

In the merging step, Scan from lowest x sides and track a running maximum x and y. If any point is dominated by the running local max_x or max_y, drop it.





Keep merging until complete.

There are $\log(n)$ levels and a linear $O(n)$ scan is done at each level.

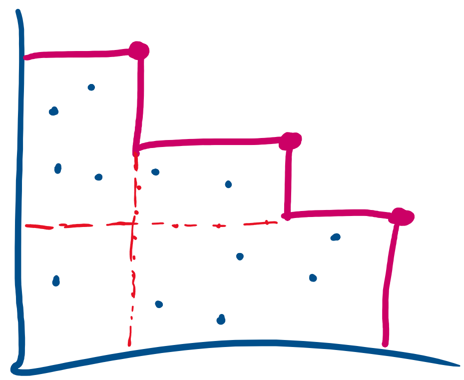
$$\boxed{O(n \cdot \log(n))}$$

---

d) To improve the algorithm in part A from $O(n \cdot \log(n))$ to $O(n \cdot \log(h))$, we can only pass points on that must be on the staircase.

Observe that the 'top' of the staircase has the smallest x and greatest y. Conversely, the 'bottom' has the largest x and smallest y.

We can track these globally to exclude more points, preserving only global extremals.



We can split from the median $O(n)$ into left and right, such that all points lie below the median in the left group and above the median on the right (on the x-axis).

Split until we have groups length ≤ 2.

When merging on a left, track the left's max_x, and drop any points where $y < max_y$, or dominated like part A.

In parallel, on the right, track the right's max_x, and drop any points where $x < max_x$, or dominated like part A.

Sort by x-coordinate and pass on the extremals, like part A.

Sorting two sorted lists is $O(n)$

Repeat until complete.

Median selection using median of medians is $O(n)$.

Scanning and sorting two sorted lists is $O(n)$.

There are $\log(h)$ levels because each merge step guarantees one extremal is found due to the max_x and max_y. If each merge step guarantees a point, there can be at most $\log(h)$ levels.

$$T(n,h) = 2n + T(\tfrac{n}{2},h) + T(\tfrac{n}{2},h)$$

$$T(n,h) = n\log h$$

$$\boxed{O(n\log h)}$$