Max Points on a Line

Core Idea:

Fix each point i as an anchor, then count how many other points share the same slope with it. The most frequent slope through i gives the maximum number of collinear points that pass through i. Take the max over all anchors.

How to represent a slope robustly (no floating-point!):

Use a reduced rational (dy, dx):

- Let $dy = y_j - y_i$, $dx = x_j - x_i$.
- Reduce by $g = \gcd(|dy|, |dx|)$: $dy /= g$, $dx /= g$.
- Normalize sign so each geometric slope has a unique key:
  - Force $dx > 0$. If $dx < 0$, flip both signs.
  - For vertical lines, set to (1, 0).
  - For horizontal lines, set to (0, 1).
  - For the zero vector (shouldn't occur since points are unique), you'd normally count duplicates - here constraints say all points are unique.

Use a hash map from (dy, dx) → count.

Per-anchor procedure:

For each anchor i:

1. Clear the slope counter map.
2. For each other point j:
   - Compute (dy, dx), reduce & normalize.
   - Increment the count for that slope.
3. The best through i is 1 + max count_on_any_slope (the +1 for the anchor itself).
4. Track a global maximum.

Micro-optimizations (optional):

- Early pruning: after processing anchors 0...i, if global_max >= n-i, you can stop (can't beat it).
- Use a compact key for the slope (e.g. pair of ints) with a custom hash; avoid

strings.