Divide & Conquer

Break up problem into nontrivial subparts
Can be solved separately

Combine the solutions to subproblems

Mergesort

Quicksort

Mergesort → Count Inversions

Closest distance

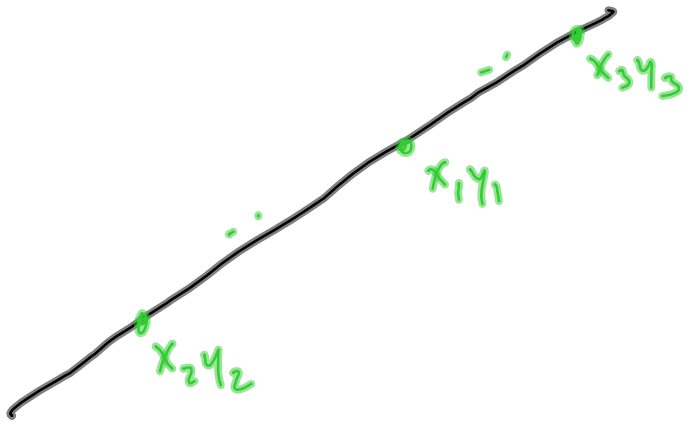
Points (x_1, y_1)

(x_2, y_2)

$\vdots$

(x_k, y_k)

on a line



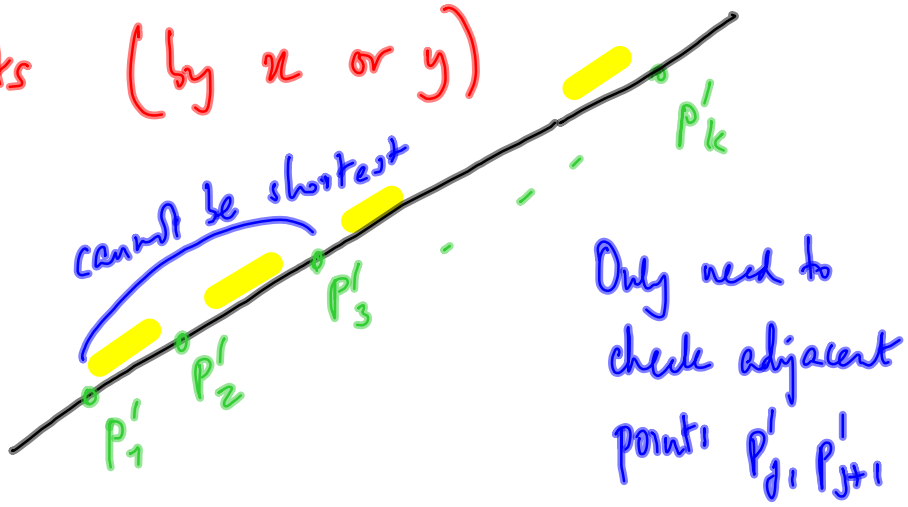
Find closest pair (x_i, y_i) & (x_j, y_j)

Brute force: $p_i = (x_i, y_i)$

Check distance $d(p_i, p_j)$ for all pairs

$O(k^2)$

Sort the points (by x or y)



Total cost

$O(k \log k)$ to sort

$O(k)$ to check each interval

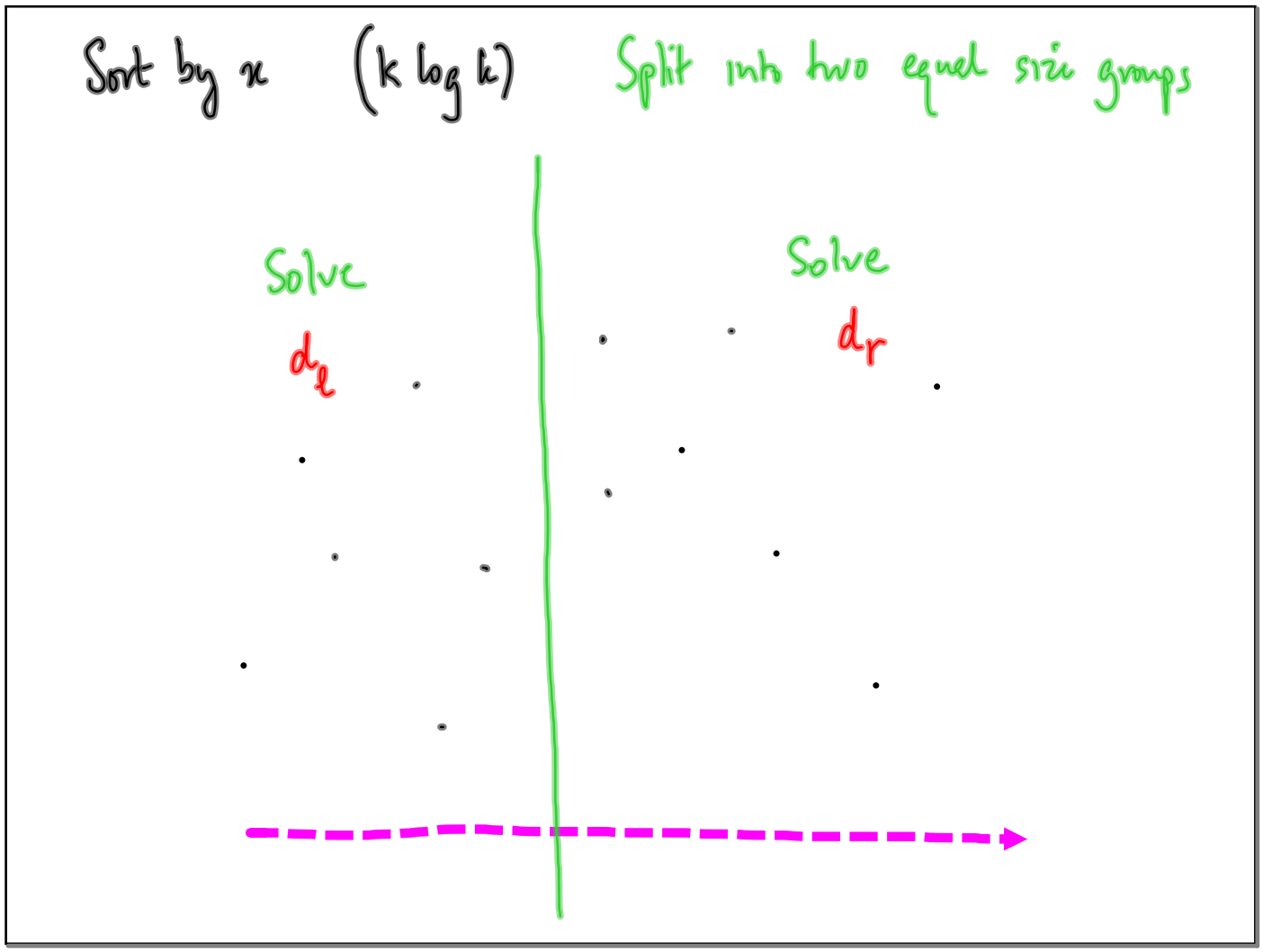
$\Rightarrow O(k \log k)$

What if the k points are not on a line?

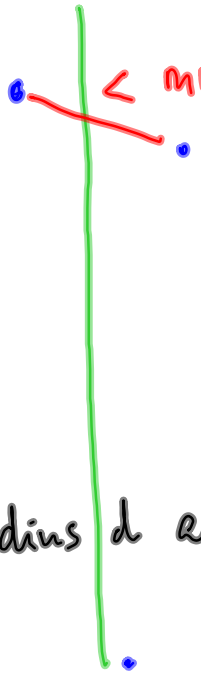
Assume no two points have same x or y coord.

Trivially $O(k^2)$

Want something better

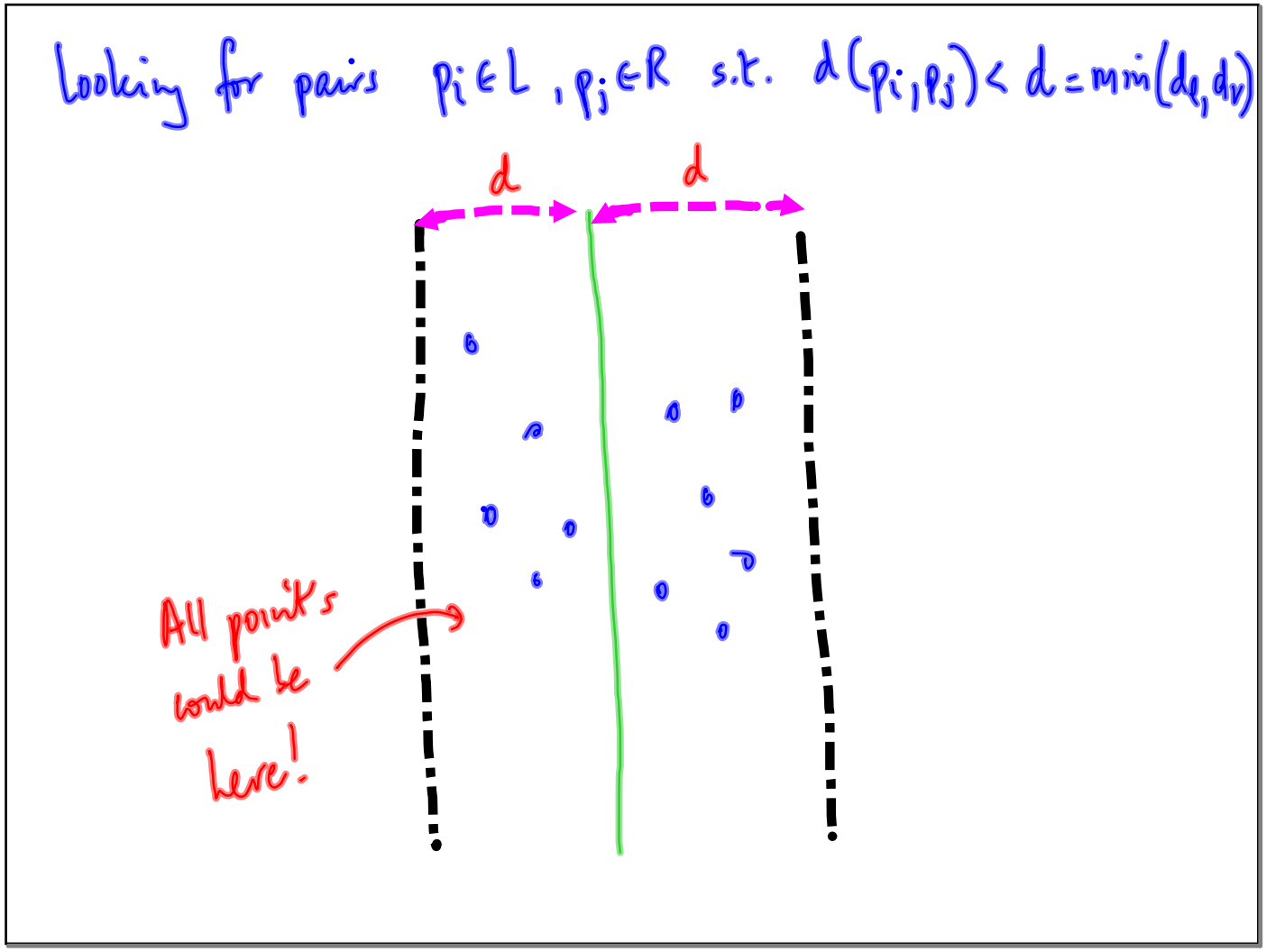

Sort by x
(k log k)
Split into two equal size groups
Solve
d_l
Solve
d_r

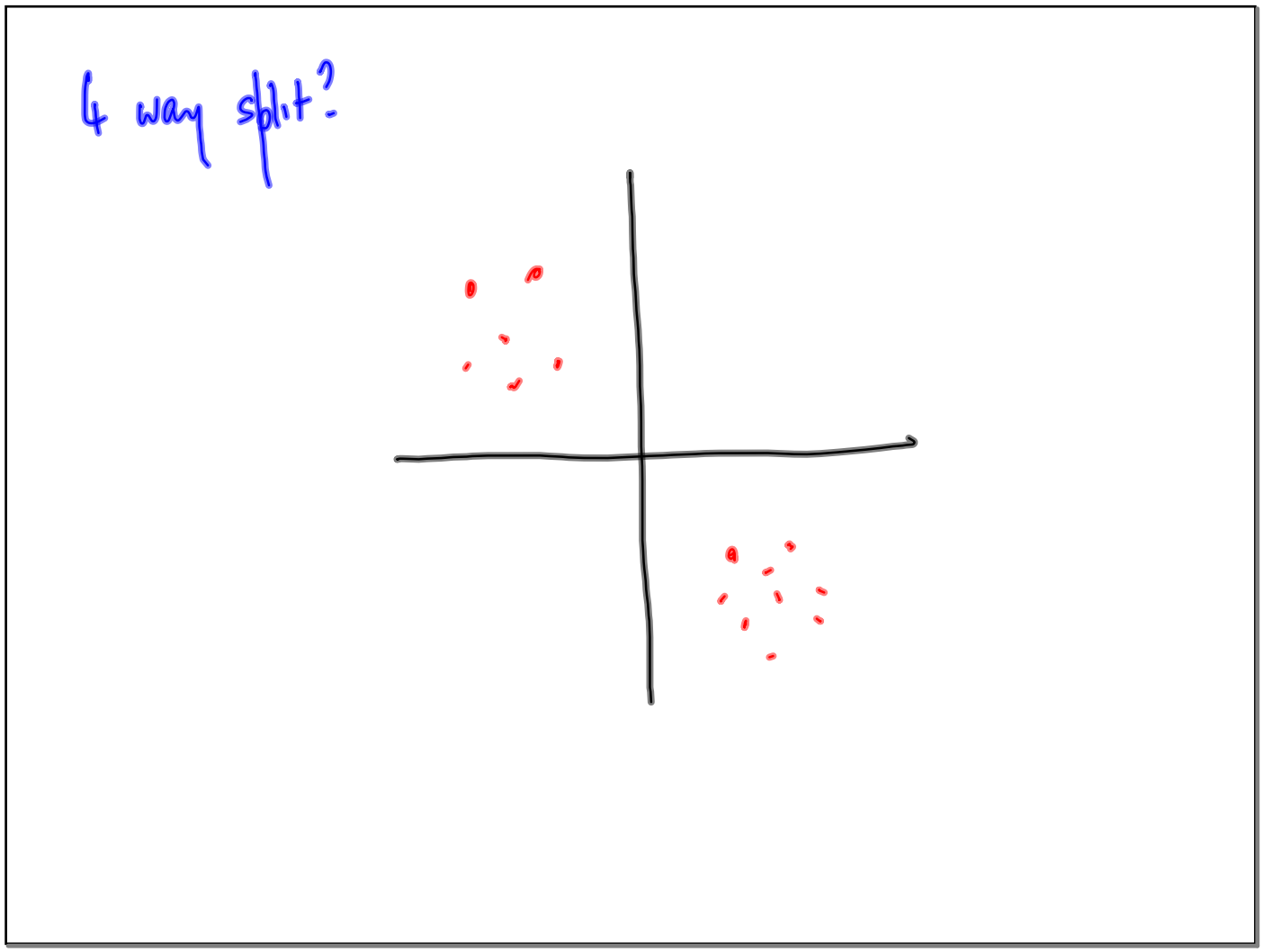
After solving the two halves, need to check points across the boundary

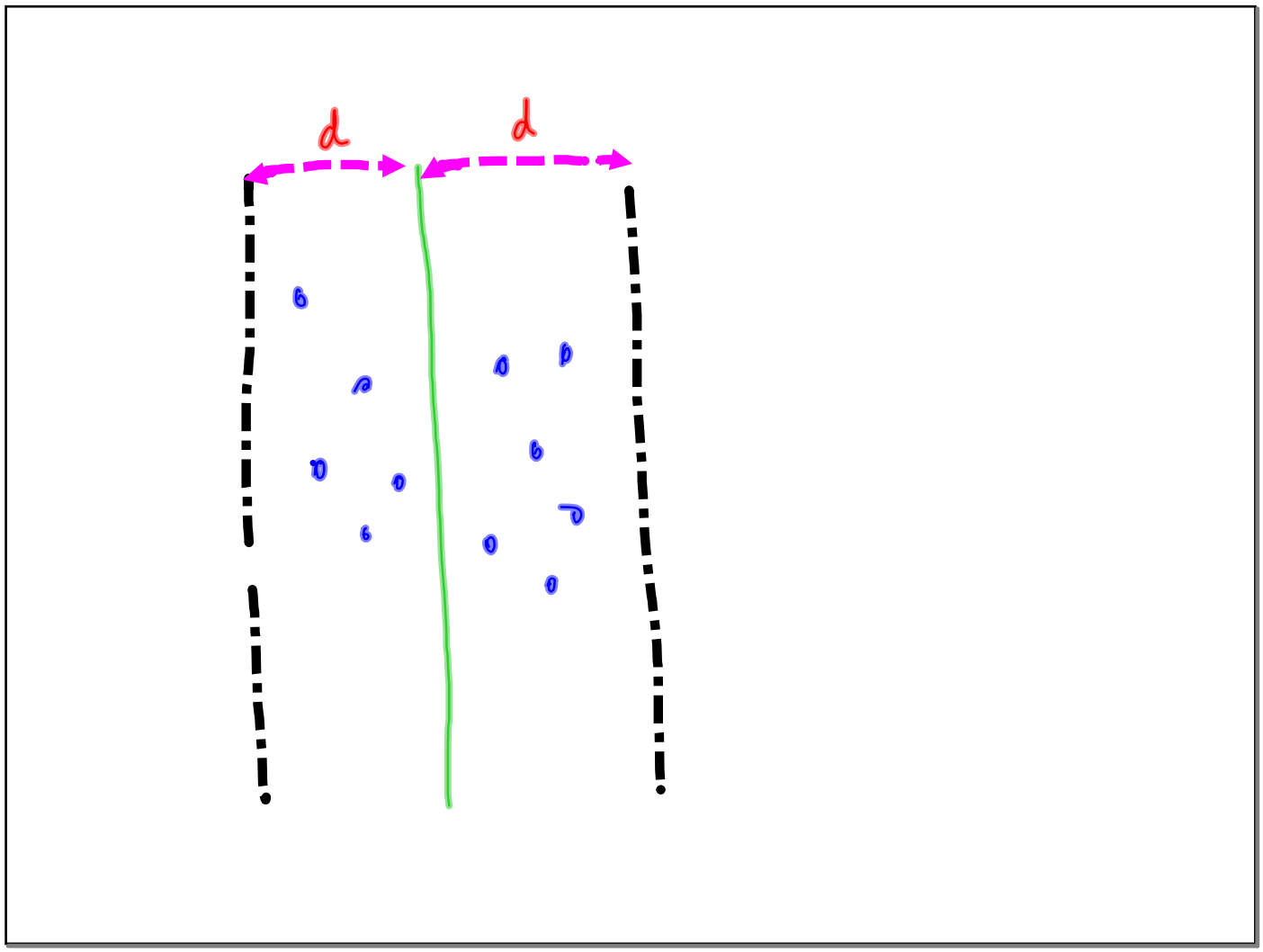


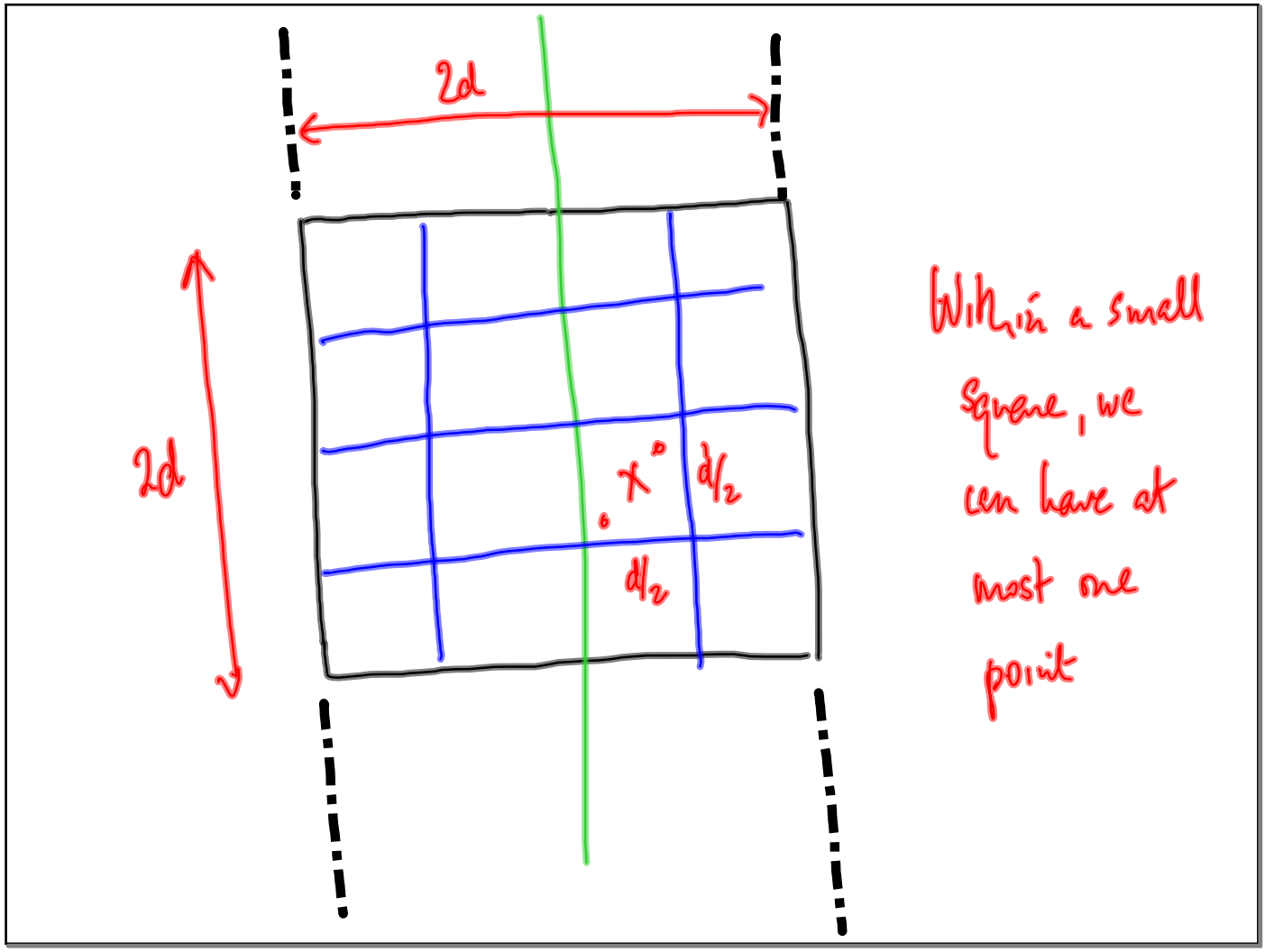
$d = \min(d_l, d_r)$

Draw a circle of radius d around all points in the "smaller" side ?


Looking for pairs $p_i \in L$, $p_j \in R$ s.t. $d(p_i, p_j) < d = \min(d_l, d_r)$
d
d
All points could be here!


4 way split?

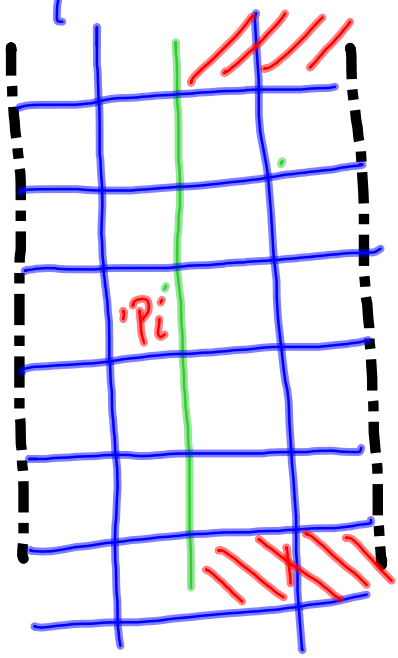

d
d


2d
2d
x
d/2
d/2
Within a small square, we can have at most one point

Suppose $p_i \in L$ lies in the zone of interest

Assume entire $-d, +d$ band is tiled by

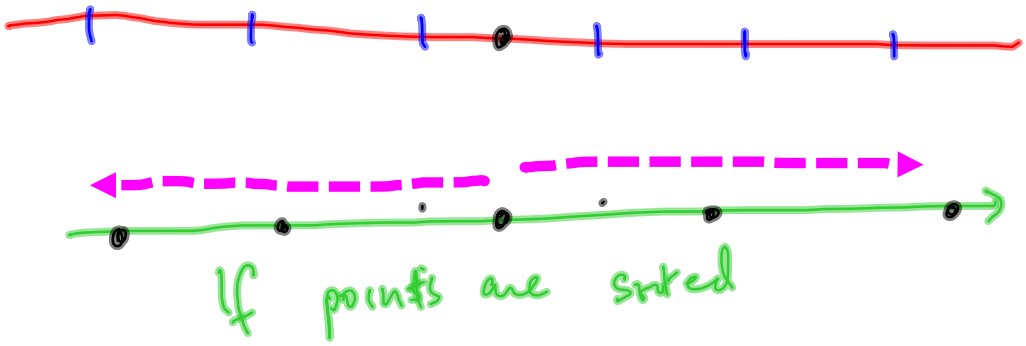
$d/2 \times d/2$ squares



Need to compare each $p_i \in L$ to at most 10 $p_j \in R$

Need to do some careful bookkeeping to ensure we do this efficiently

In 1d

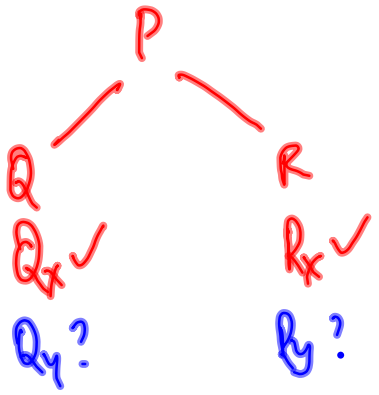


Initially

points P

sort by x P_x

sort by y P_y



Need to maintain back & forth pointers across these sorted lists

Multiplication