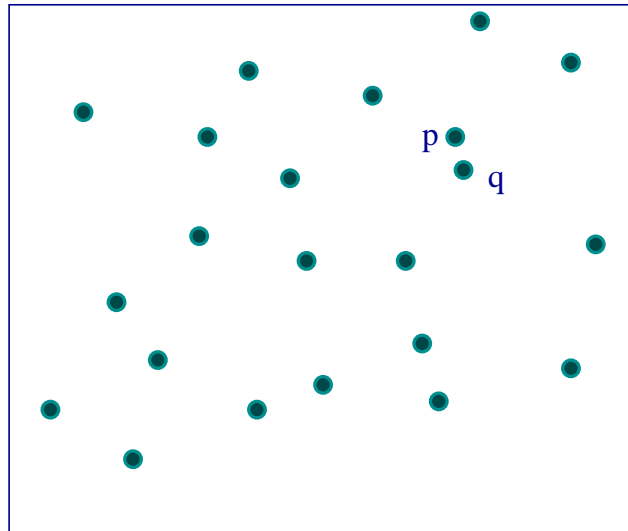


# Closest Pair Problem

---

- Given  $n$  points in  $d$ -dimensions, find two whose mutual distance is smallest.
- Fundamental problem in many applications as well as a key step in many algorithms.

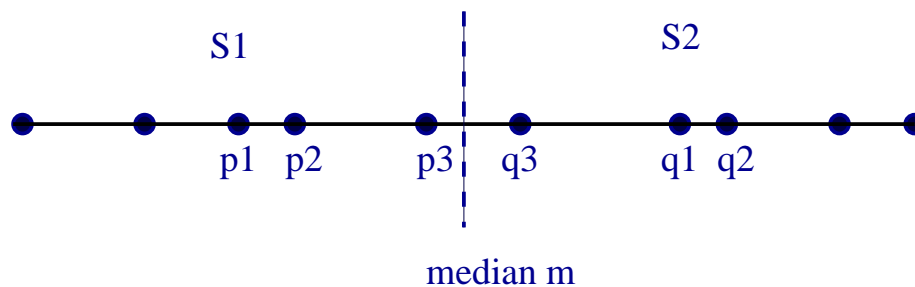


- A naive algorithm takes  $O(dn^2)$  time.
- Element uniqueness reduces to Closest Pair, so  $\Omega(n \log n)$  lower bound.
- We will develop a divide-and-conquer based  $O(n \log n)$  algorithm; dimension  $d$  assumed constant.

# 1-Dimension Problem

---

- 1D problem can be solved in  $O(n \log n)$  via sorting.
- Sorting, however, does not generalize to higher dimensions. So, let's develop a divide-and-conquer for 1D.
- Divide the points  $S$  into two sets  $S_1, S_2$  by some  $x$ -coordinate so that  $p < q$  for all  $p \in S_1$  and  $q \in S_2$ .
- Recursively compute closest pair  $(p_1, p_2)$  in  $S_1$  and  $(q_1, q_2)$  in  $S_2$ .

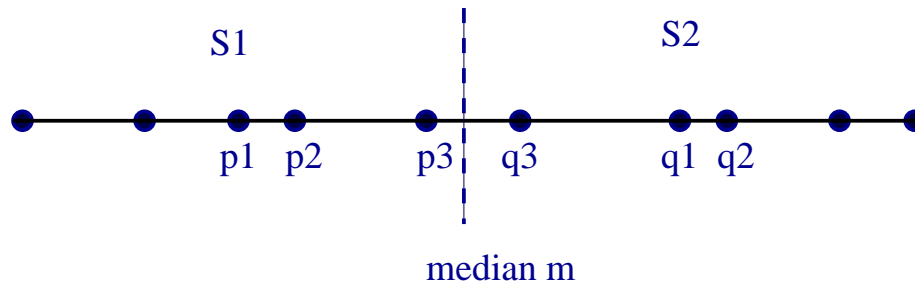


- Let  $\delta$  be the smallest separation found so far:

$$\delta = \min(|p_2 - p_1|, |q_2 - q_1|)$$

# 1D Divide & Conquer

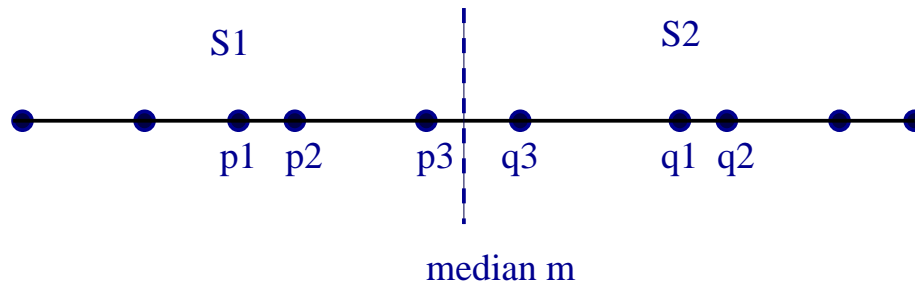
---



- The closest pair is  $\{p_1, p_2\}$ , or  $\{q_1, q_2\}$ , or some  $\{p_3, q_3\}$  where  $p_3 \in S_1$  and  $q_3 \in S_2$ .
- **Key Observation:** If  $m$  is the dividing coordinate, then  $p_3, q_3$  must be within  $\delta$  of  $m$ .
- In 1D,  $p_3$  must be the rightmost point of  $S_1$  and  $q_3$  the leftmost point of  $S_2$ , but these notions do not generalize to higher dimensions.
- How many points of  $S_1$  can lie in the interval  $(m - \delta, m]$ ?
- By definition of  $\delta$ , at most one. Same holds for  $S_2$ .

# 1D Divide & Conquer

---

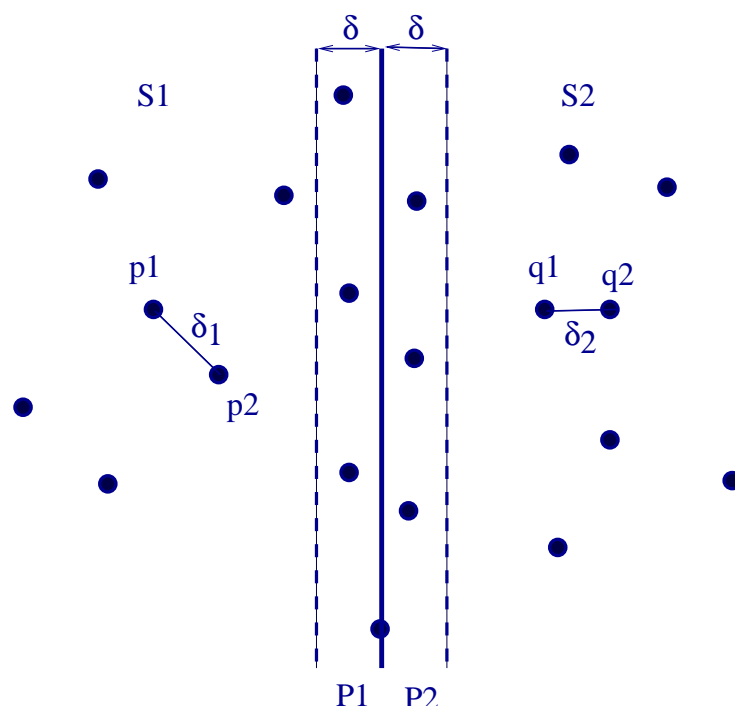


- **Closest-Pair** ( $S$ ).
- If  $|S| = 1$ , output  $\delta = \infty$ .  
If  $|S| = 2$ , output  $\delta = |p_2 - p_1|$ .  
Otherwise, do the following steps:
  1. Let  $m = \text{median}(S)$ .
  2. Divide  $S$  into  $S_1, S_2$  at  $m$ .
  3.  $\delta_1 = \text{Closest-Pair}(S_1)$ .
  4.  $\delta_2 = \text{Closest-Pair}(S_2)$ .
  5.  $\delta_{12}$  is minimum distance across the cut.
  6. Return  $\delta = \min(\delta_1, \delta_2, \delta_{12})$ .
- Recurrence is  $T(n) = 2T(n/2) + O(n)$ , which solves to  $T(n) = O(n \log n)$ .

# 2-D Closest Pair

---

- We partition  $S$  into  $S_1, S_2$  by vertical line  $\ell$  defined by median  $x$ -coordinate in  $S$ .
- Recursively compute closest pair distances  $\delta_1$  and  $\delta_2$ . Set  $\delta = \min(\delta_1, \delta_2)$ .
- Now compute the closest pair with one point each in  $S_1$  and  $S_2$ .

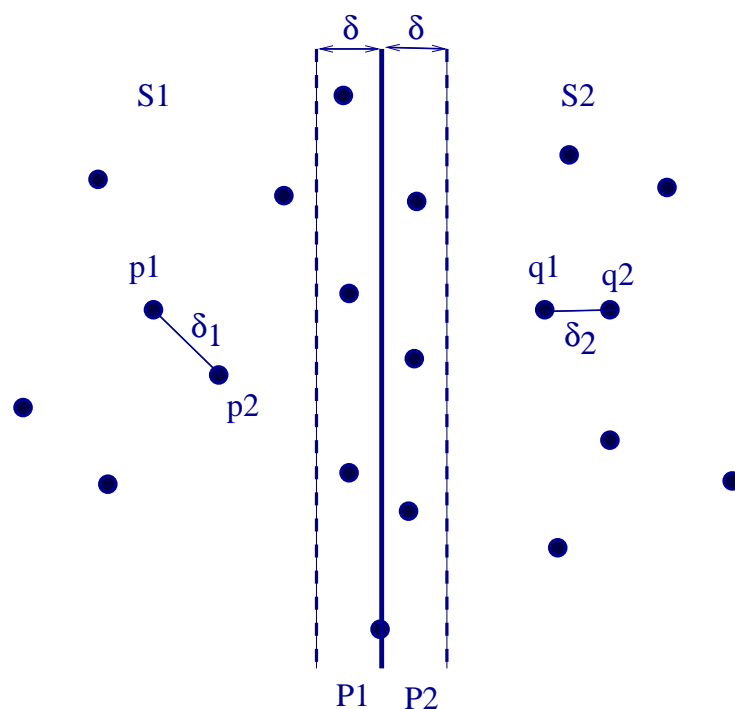


- In each candidate pair  $(p, q)$ , where  $p \in S_1$  and  $q \in S_2$ , the points  $p, q$  must both lie within  $\delta$  of  $\ell$ .

# 2-D Closest Pair

---

- At this point, complications arise, which weren't present in 1D. It's entirely possible that all  $n/2$  points of  $S_1$  (and  $S_2$ ) lie within  $\delta$  of  $\ell$ .

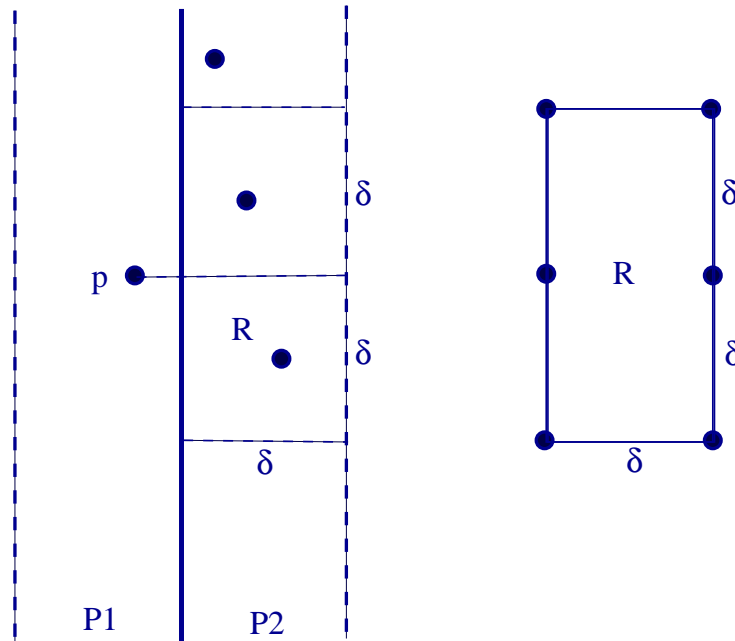


- Naively, this would require  $n^2/4$  calculations.
- We show that points in  $P_1, P_2$  ( $\delta$  strip around  $\ell$ ) have a special structure, and solve the conquer step faster.

# Conquer Step

---

- Consider a point  $p \in S_1$ . All points of  $S_2$  within distance  $\delta$  of  $p$  must lie in a  $\delta \times 2\delta$  rectangle  $R$ .

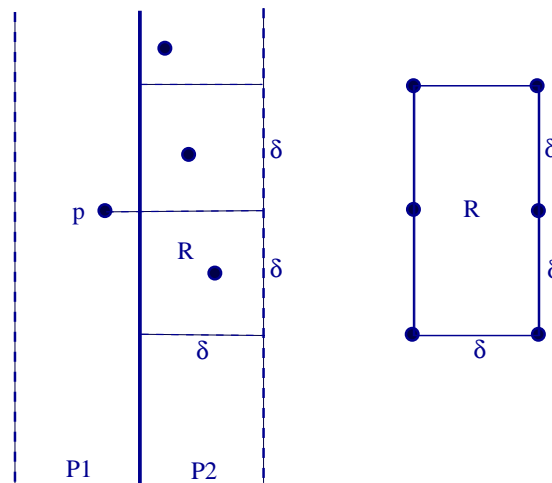


- How many points can be inside  $R$  if each pair is at least  $\delta$  apart?
- In 2D, this number is at most 6!
- So, we only need to perform  $6 \times n/2$  distance comparisons!
- We don't have an  $O(n \log n)$  time algorithm yet. Why?

# Conquer Step Pairs

---

- In order to determine at most 6 potential mates of  $p$ , project  $p$  and all points of  $P_2$  onto line  $\ell$ .



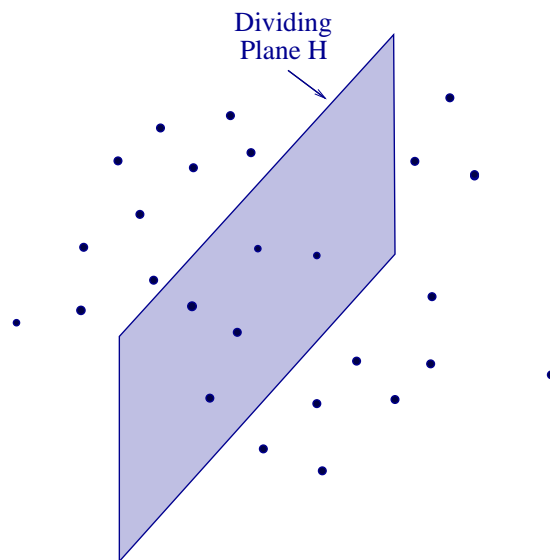
- Pick out points whose projection is within  $\delta$  of  $p$ ; at most six.
- We can do this for all  $p$ , by walking sorted lists of  $P_1$  and  $P_2$ , in total  $O(n)$  time.
- The sorted lists for  $P_1, P_2$  can be obtained from pre-sorting of  $S_1, S_2$ .
- Final recurrence is  $T(n) = 2T(n/2) + O(n)$ , which solves to  $T(n) = O(n \log n)$ .



# $d$ -Dimensional Closest Pair

---

- Two key features of the divide and conquer strategy are these:
  1. The step where subproblems are combined takes place in one lower dimension.
  2. The subproblems in the combine step satisfy a sparsity condition.
  3. **Sparsity Condition:** Any cube with side length  $2\delta$  contains  $O(1)$  points of  $S$ .
  4. Note that the original problem does not necessarily have this condition.



# The Sparse Problem

---

- Given  $n$  points with  $\delta$ -sparsity condition, find all pairs within distance  $\leq \delta$ .
- Divide the set into  $S_1, S_2$  by a median plane  $H$ . Recursively solve the problem in two halves.
- Project all points lying within  $\delta$  thick slab around  $H$  onto  $H$ . Call this set  $S'$ .
- $S'$  inherits the  $\delta$ -sparsity condition. Why?.
- Recursively solve the problem for  $S'$  in  $d - 1$  space.
- The algorithm satisfies the recurrence

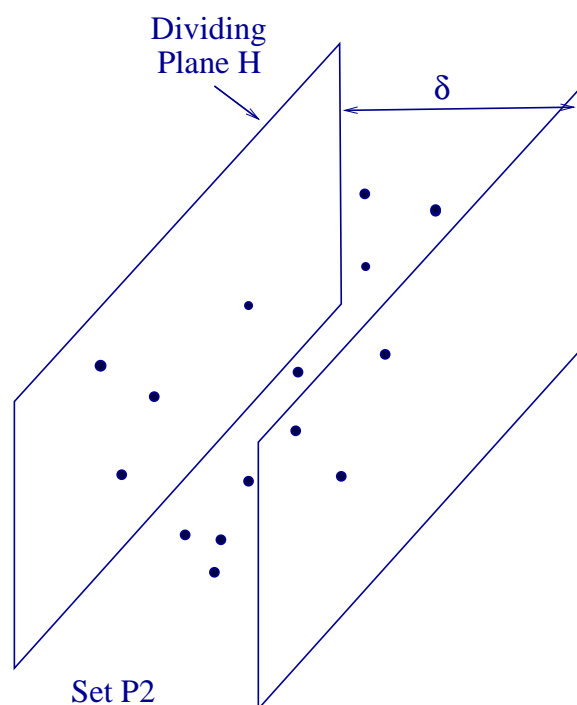
$$U(n, d) = 2U(n/2, d) + U(n, d - 1) + O(n).$$

which solves to  $U(n, d) = O(n(\log n)^{d-1})$ .

# Getting Sparsity

---

- Recall that divide and conquer algorithm solves the left and right half problems recursively.
- The sparsity holds for the merge problem, which concerns points within  $\delta$  thick slab around  $H$ .

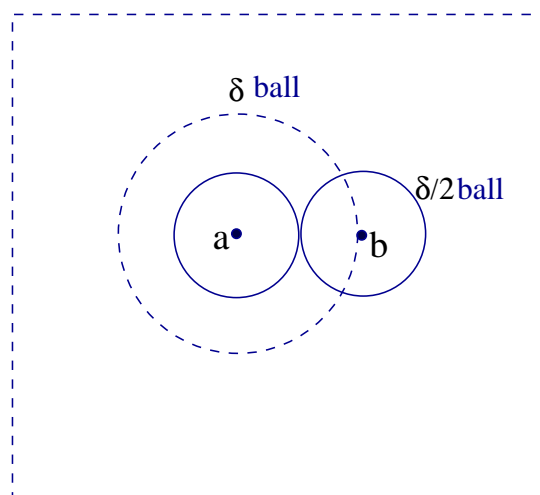


- If  $S$  is a set where inter-point distance is at least  $\delta$ , then the  $\delta$ -cube centered at  $p$  contains at most a constant number of points of  $S$ , depending on  $d$ .

# Proof of Sparsity

---

- Let  $C$  be the  $\delta$ -cube centered at  $p$ . Let  $L$  be the set of points in  $C$ .
- Imagine placing a ball of radius  $\delta/2$  around each point of  $L$ .
- No two balls can intersect. Why?
- The volume of cube  $C$  is  $(2\delta)^d$ .
- The volume of each ball is  $\frac{1}{c_d}(\delta/2)^d$ , for a constant  $c_d$ .
- Thus, the maximum number of balls, or points, is at most  $c_d 4^d$ , which is  $O(1)$ .



# Closest Pair Algorithm

---

- Divide the input  $S$  into  $S_1, S_2$  by the median hyperplane normal to some axis.
- Recursively compute  $\delta_1, \delta_2$  for  $S_1, S_2$ . Set  $\delta = \min(\delta_1, \delta_2)$ .
- Let  $S'$  be the set of points that are within  $\delta$  of  $H$ , **projected onto  $H$** .
- Use the  $\delta$ -sparsity condition to recursively examine all pairs in  $S'$ —there are only  $O(n)$  pairs.
- The recurrence for the final algorithm is:

$$\begin{aligned} T(n, d) &= 2T(n/2, d) + U(n, d-1) + O(n) \\ &= 2T(n/2, d) + O(n(\log n)^{d-2}) + O(n) \\ &= O(n(\log n)^{d-1}). \end{aligned}$$

# Improving the Algorithm

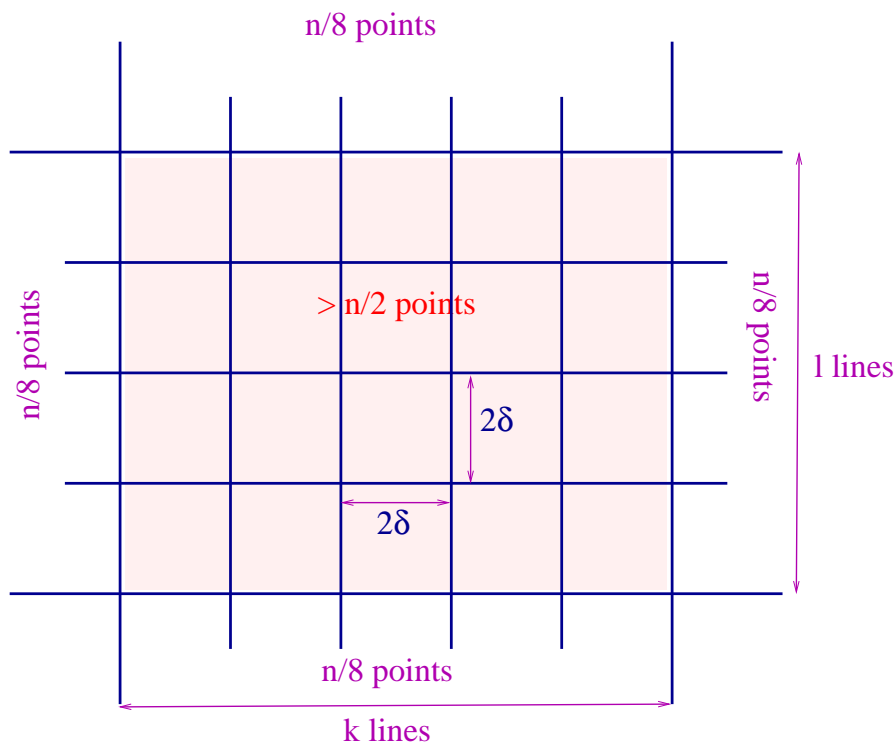
---

- If we could show that the problem size in the conquer step is  $m \leq n/(\log n)^{d-2}$ , then  $U(m, d-1) = O(m(\log m)^{d-2}) = O(n)$ .
- This would give final recurrence  $T(n, d) = 2T(n/2, d) + O(n) + O(n)$ , which solves to  $O(n \log n)$ .
- **Theorem:** Given a set  $S$  with  $\delta$ -sparsity, there exists a hyperplane  $H$  normal to some axis such that
  1.  $|S_1|, |S_2| \geq n/4d$ .
  2. Number of points within  $\delta$  of  $H$  is  $O\left(\frac{n}{(\log n)^{d-2}}\right)$ .
  3.  $H$  can be found in  $O(n)$  time.

# Sparse Hyperplane

---

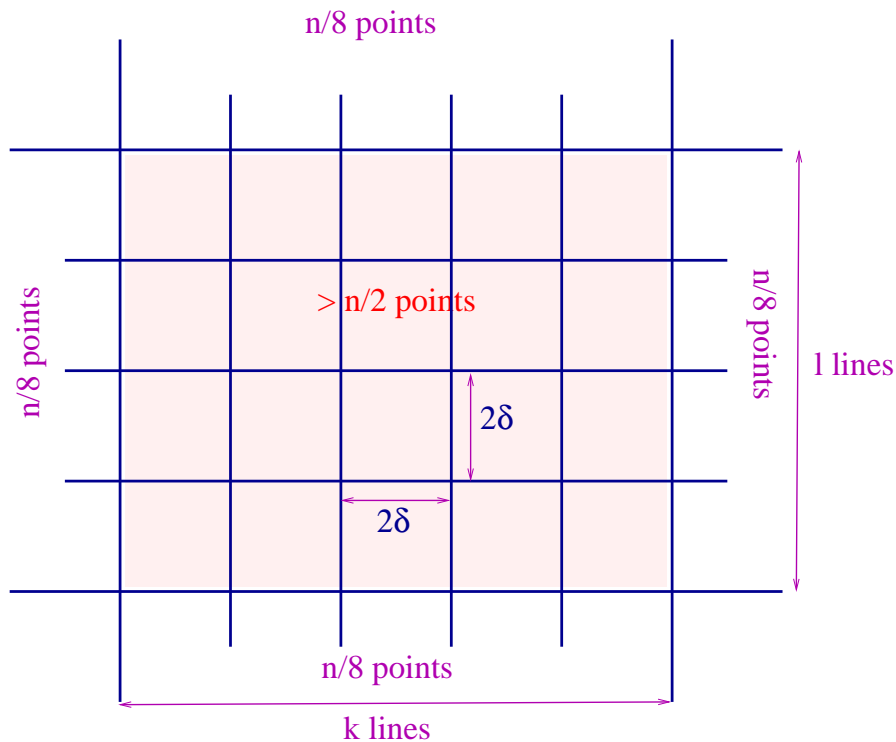
- We prove the theorem for 2D. Show there is a line with  $\alpha\sqrt{n}$  points within  $\delta$  of it, for some constant  $\alpha$ .
- For contradiction, assume no such line exists.
- Partition the plane by placing vertical lines at distance  $2\delta$  from each other, where  $n/8$  points to the left of leftmost line, and right of rightmost line.



# Sparse Hyperplane

---

- If there are  $k$  slabs, we have  $k\alpha\sqrt{n} \leq 3n/4$ , which gives  $k \leq \frac{3}{4\alpha}\sqrt{n}$ .

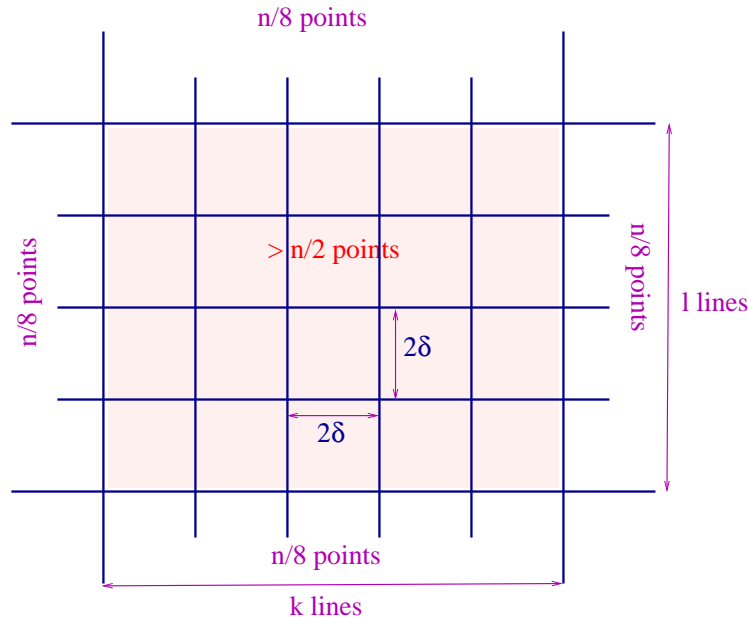


- Similarly, if there is no horizontal line with desired properties, we get  $l \leq \frac{3}{4\alpha}\sqrt{n}$ .
- By sparsity, number of points in any  $2\delta$  cell is some constant  $c$ .



# Sparse Hyperplane

---



- This gives that the num. of points inside all the slabs is at most  $ckl$ , which is at most  $\left(\frac{3}{4\alpha}\right)^2 cn$ .
- Since there are  $\geq n/2$  points inside the slabs, this is a contradiction if we choose  $\alpha \geq \frac{\sqrt{18c}}{4}$ .
- So, one of these  $k$  vertical or  $l$  horizontal lines must satisfy the desired properties.
- Since we know  $\delta$ , we can check these  $k + l$  lines and choose the correct one in  $O(n)$  time.

# Optimal Algorithm

---

- Actually we can start the algorithm with such a hyperplane.
- The divide and conquer algorithm now satisfies the recurrence
$$T(n, d) = 2T(n/2, d) + U(m, d - 1) + O(n).$$
- By new sparsity claim,  $m \leq n/(\log n)^{d-2}$ , and so  $U(m, d - 1) = O(m(\log m)^{d-2}) = O(n)$ .
- Thus,  $T(n, d) = 2T(n/2, d) + O(n) + O(n)$ , which solves to  $O(n \log n)$ .
- Solves the Closest Pair problem in fixed  $d$  in optimal  $O(n \log n)$  time.