Closest Pairs

Closest Pair

Problem:

Given set of pts $P = \{p_1, \dots, p_n\} \subseteq \mathbb{R}^2$

output: $p_a, p_b \in P$ with smallest distance, i.e.

$$\|p_a - p_b\| = \min \{ \|p_i - p_j\| : p_i, p_j \in P, p_i \neq p_j \}$$

"closest pair" (usually assume uniqueness)

Context: consider in 1D, solution $\in O(n \log n)$

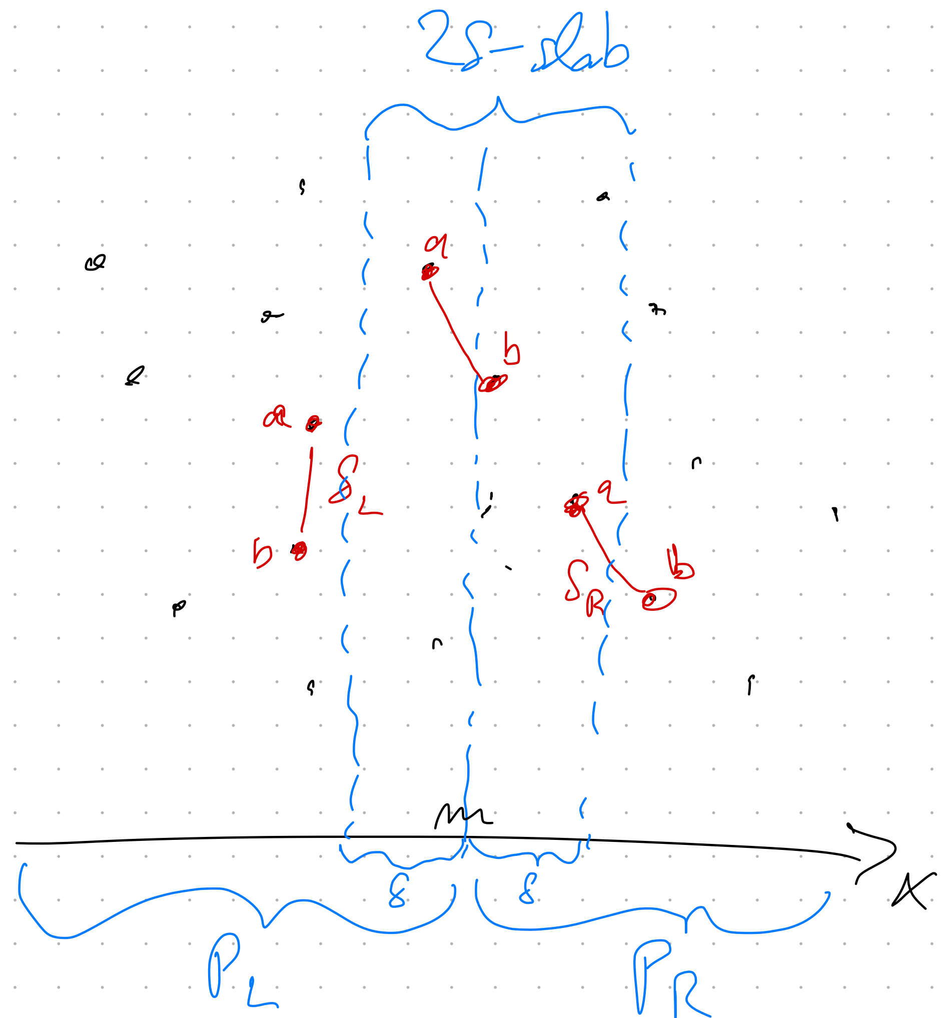
Naive algo: $O(n^2)$

Better algo:

1) Sort P along x, partition along median m
   $\rightarrow P_L, P_R$

2) Recursion with $P_L, P_R \rightarrow \delta_L, \delta_R$

3) $\delta := \min(\delta_L, \delta_R)$



Consider "2δ- slab" around median

$P' = \{ p \mid p_x \in [m - \delta, m + \delta] \} \subseteq P$

find closest pair in $P' \to \delta_M$ $\leftarrow$

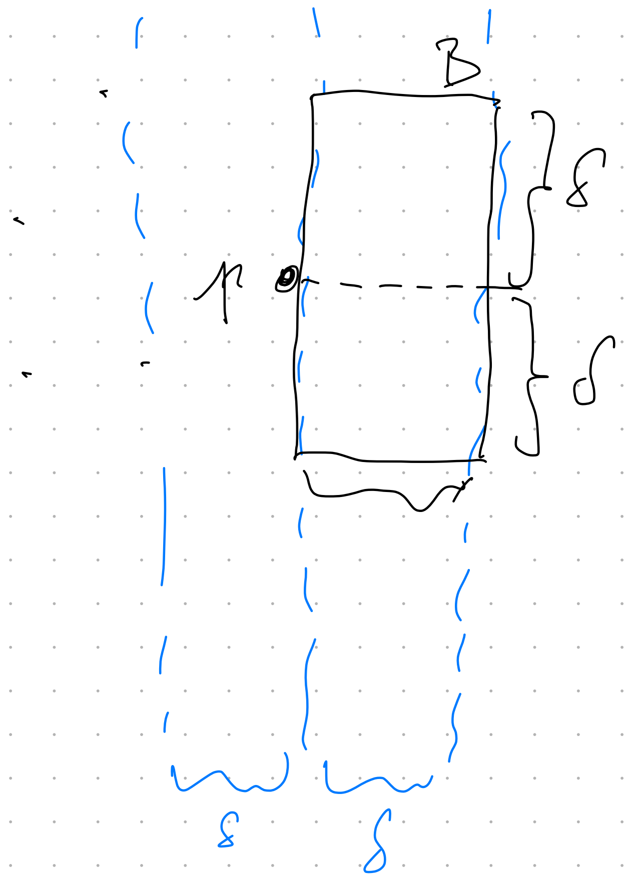
Output: $\min(\delta_L, \delta_R, \delta_M)$ together pts

Details about step 3:

Sort P' along y [can be optimized away]

iterate over P' bottom up:

for each p, consider only pts inside B



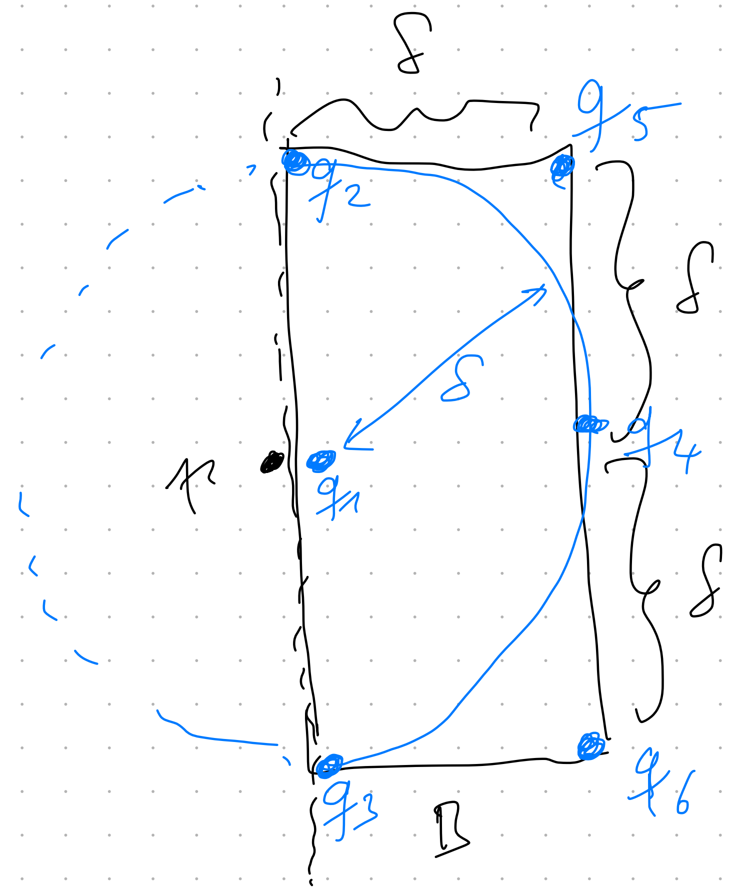
Claim: B can contain only 6 pts max

$\Rightarrow$ phase 3 $\in O(\underbrace{n \log n}_{\text{sorting}}) + O(\underbrace{n}_{\text{iteration over } P'})$

Proof of claim:

1) worst-case: $q_1 \in P$ very close to p
2) other q's must lie outside ball around q_1 with radius δ $\rightarrow q_2, q_3, q_4$
3) other pts q must have dist δ away from $q_1, \dots, q_4$ $\rightarrow q_5, q_6$

$\Rightarrow$ claim



Complexity:

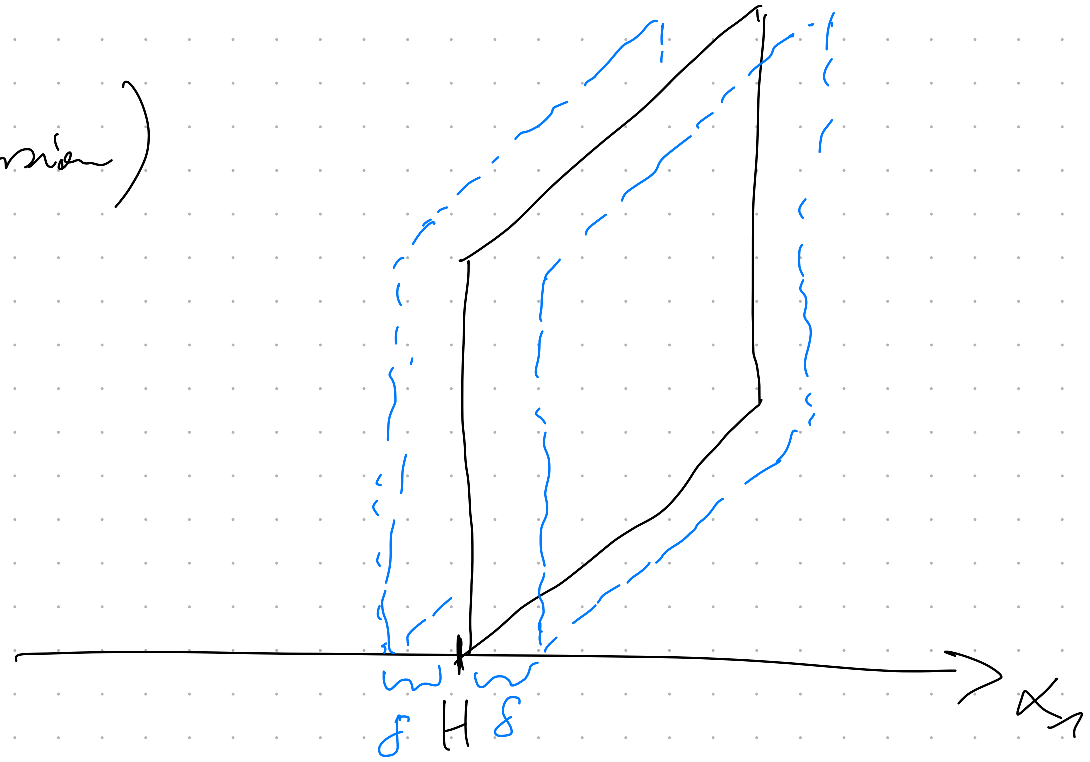
$$T(n) = \underbrace{2T\left(\frac{n}{2}\right)}_{\text{step 2}} + \underbrace{O(n \log n)}_{\text{step 1}} + \underbrace{O(n \log n)}_{\text{step 3}}$$

$$= O(n \log^{2} n)$$

Closest Pair in d-dim space

Algo:

1) Sort P along x_1, find median plane H, partition P into P_L, P_R $\rightarrow$ distance δ (recursion)
2) Find all pts $\in P$ within 2δ-slab around H, project onto H (set x_1-coord $= H$) $\rightarrow P'$
3) Find closest pair in P' ($(d-1)$-dim. version!)



Recurrence:

$$T(n,d) = \overbrace{2T\left(\tfrac{n}{2},d\right)}^{\text{step 1}} + \overbrace{O(n)}^{2} + \overbrace{U(n,d-1)}^{3}$$

claim (for now): $U(n,d-1) = O\left(n \log^{d-2} n\right)$

Then

$$T(n,d) = \underline{2}T\left(\tfrac{n}{2},d\right) + O(n) + O\left(n \log^{d-2} n\right)$$

$$= 4\,T\left(\frac{n}{4}, d\right) + 8\cdot O\left(\frac{n}{8}\right) + 8\cdot O\left(\frac{n}{8}\log^{d-2}\frac{n}{2}\right)$$

$$+ O(n) + O(n \log^{d-2} n)$$

$$= \dots \quad \Big\} \log n$$

$$= n\cdot T(1,d) + \log n \cdot O(n) + \log n \cdot O(n \log^{d-2} n)$$

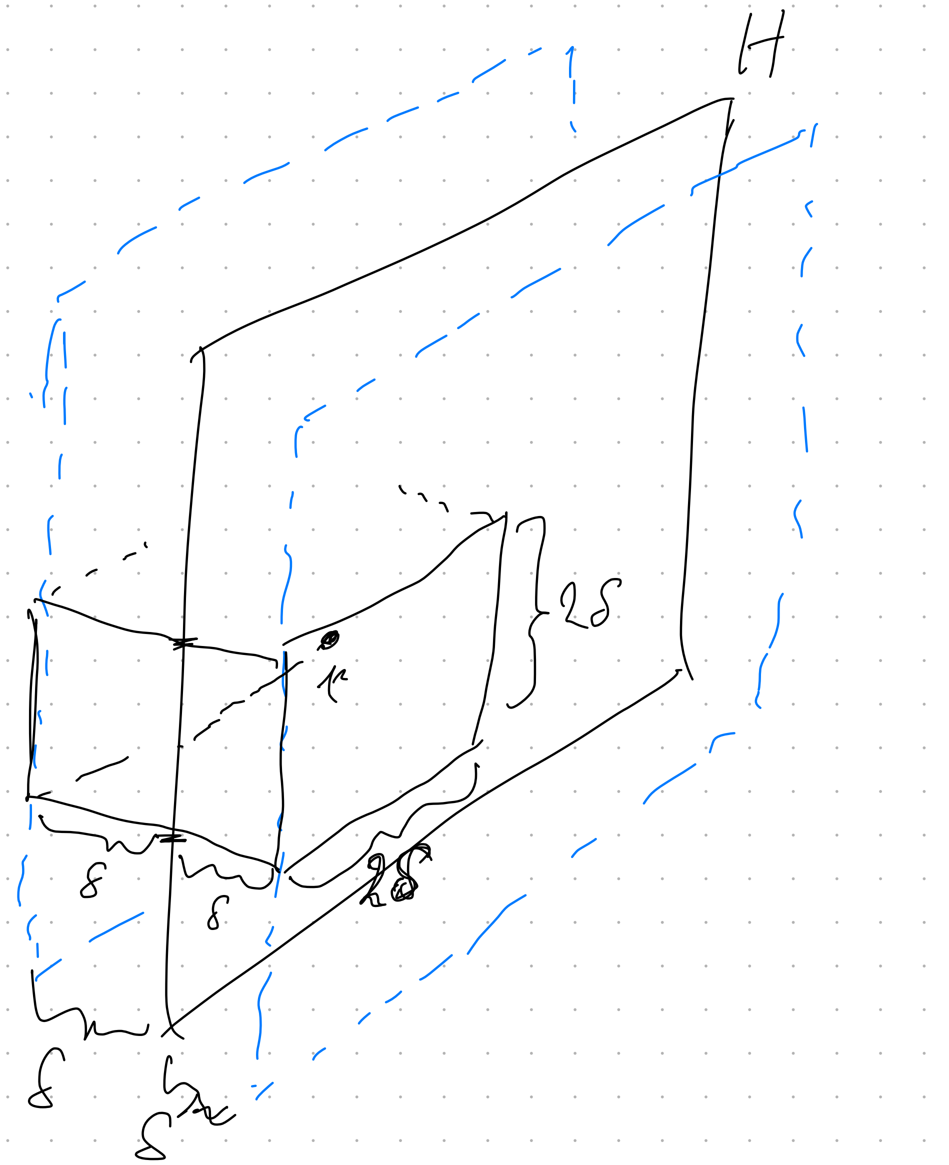
$$= O(n \log^{d-1} n)$$

Observation:

pts $\in P'$ are "δ-sparse".

Consider any $p \in P'$,

Then B centered around p with size 2δ can contain only $O(4^d)$ many pts.



Proof:

Let $P(B)$ = pts inside B

Each pt $\in P(B) \cap P_R$ has empty ball of radius $\frac{\delta}{2}$.

$$\mathrm{Vol}(B) = (2\delta)^d$$

$$\mathrm{Vol}(\text{ball}) = c_d \cdot \left(\frac{\delta}{2}\right)^d$$

$$\#\text{ balls} \le \frac{\mathrm{Vol}(B)}{\mathrm{Vol}(\text{ball})} = \frac{(2\cancel{\delta})^d}{c_d \left(\frac{\cancel{\delta}}{2}\right)^d} = 4^d \cdot c_d' \in O(4^d)$$

(non-overlapping!)

Step 3: Find all pairs in P', $\underbrace{\text{which}}$ is δ-sparse, with distance $\le \delta$

important difference to general closest pairs algo!

Algo:

1) Find median plane H, partition $\Rightarrow P_1'$, P_2', solve recursively

2) Project pts in δ-slab around H onto $H \Rightarrow P''$,

$\Rightarrow$ recursion in $(d-1)$ dimensions

Note: P'' is δ-sparse, too!

(same argument as before)

Recurrence: $U(n, d) = 2 \cdot \underbrace{U\left(\frac{n}{2}, d\right)}_{\text{step 1}} + \underbrace{U(n, d-1) + O(n)}_{2}$

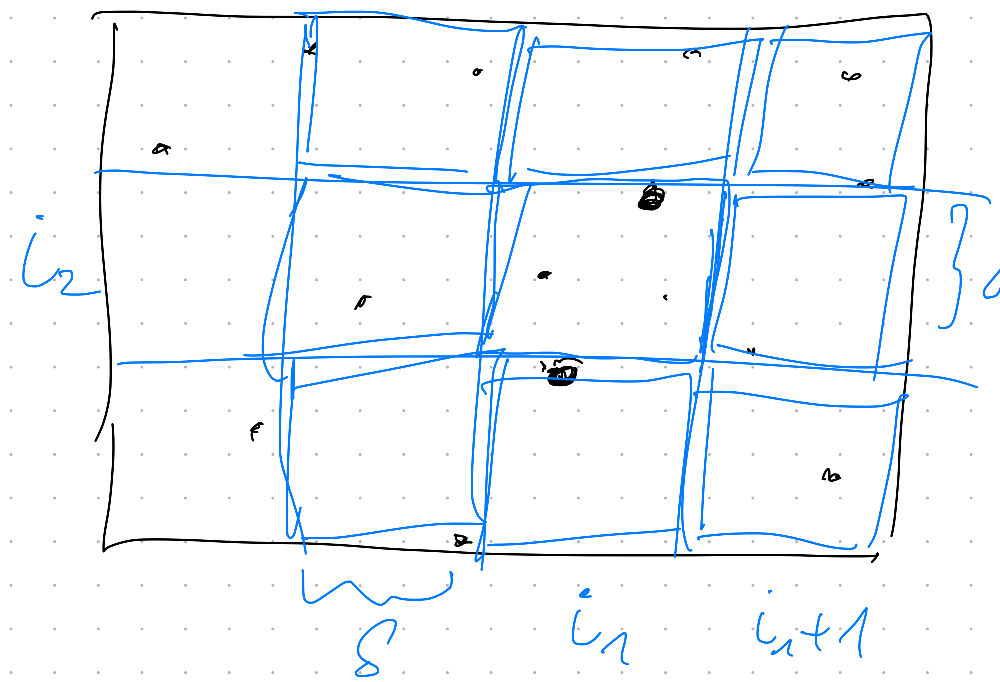
$= \ldots$

$= O(n \log^{d-2} n)$

Probabilistic algorithm in d-dim

Brute force: $O(d\,n^2)$

Idea: imagine P overlaid by "background grid" with cell ~~size~~ size δ.



(In practice: store in hash table!)

$\text{hash}(i_1, \dots, i_d) \to \text{slot}$

Algo:

Randomize $P \to p_1, \dots, p_n$

Set $\delta = \| p_1 - p_2 \|$

Fill hash table using background grid with size δ

for $i = 3, \dots, n$:

check cell of p_i and neighboring cells (*)

if new smaller dist $< \delta$:

update δ

rehash P (**)

Step $(*) \in O(1)$

Note: in $(*)$ only need to consider distances $\|p_i - p_j\|$, with $j \leq i$!

$\Rightarrow$ hash table only has to store $p_1, \dots, p_i$

$\Rightarrow$ cost (rehashing) $\in O(i)$ $\rightarrow$ step $(**)$

Expected running time:

$$T(n) = \sum_{i=2}^{n} \underbrace{\text{Prob}(\text{rehash at step } i)}_{} \cdot \underbrace{\text{Cost}(\text{rehash at } i)}_{}$$

$$= \sum_{2}^{n} \frac{2}{i} \cdot i = O(n \cdot d)$$