# CS 341: Algorithms
# Lec 04: Divide and Conquer (part 2)

Armin Jamshidpey     Collin Roberts

Based on lecture notes by Éric Schost

David R. Cheriton School of Computer Science, University of Waterloo

Winter 2025

# Closest pairs

**Goal:** given $n$ points $(x_i, y_i)$ in the plane, find a pair $(i, j)$ that minimizes the distance

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Equivalent to minimize

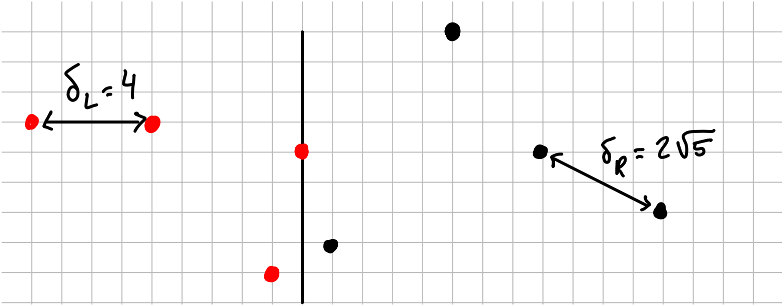
$$d_{i,j}^2 = (x_i - x_j)^2 + (y_i - y_j)^2$$

**Assumption:** all $x_i$'s are pairwise distinct

# Divide-and-conquer

**Idea:** separate the points into two halves $L, R$ at the median $x$-value

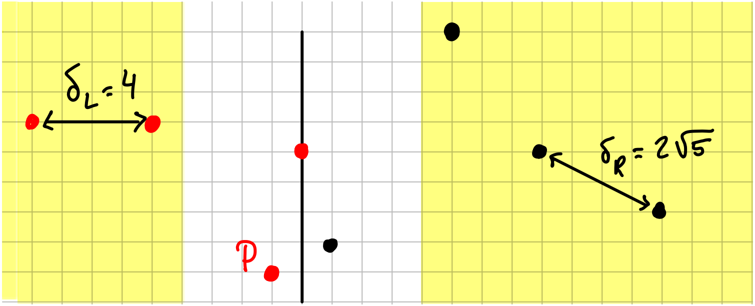
- $L =$ all $n/2$ points with $x \leq x_{\text{median}}$
- $R =$ all $n/2$ points with $x > x_{\text{median}}$
- the closest pair is either between points in $L$ (done), or between points in $R$ (done), or transverse (one in $L$, one in $R$)



# Finding the shortest transverse distance

Set $\delta = \min(\delta_L, \delta_R)$

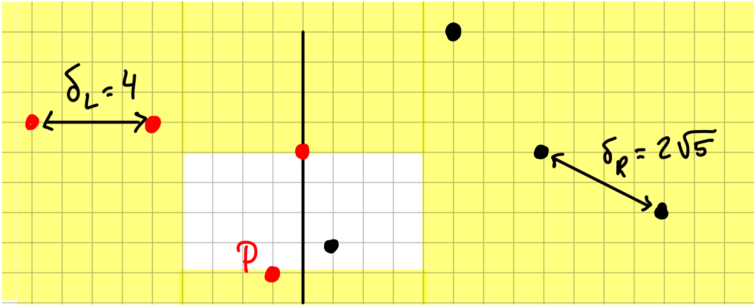
- We only need to consider transverse pairs $(P, Q)$ with $\mathbf{dist}(P, R) \leq \delta$ and $\mathbf{dist}(Q, L) \leq \delta$.



# Finding the shortest transverse distance

Set $\delta = \min(\delta_L, \delta_R)$

- For any $P = (x_P, y_P)$, enough to look at points with $y_P \leq y < y_P + \delta$



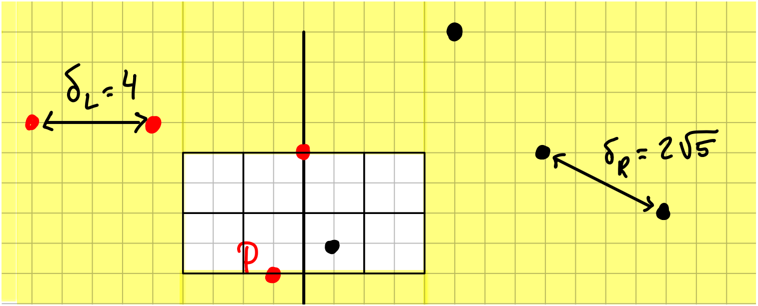

So it is enough to check distances $d(P, Q)$ for $Q$ in the rectangle.

# How many points in the rectangle?

**Claim**

There are at most **8** points from our initial set (including $P$) in the rectangle.

**Proof.** Cover the rectangle with **8** squares of side length $\delta/2$




Squares on the left only contain points from $L$, squares on the right only contain points from $R$.

Consequence: in each square, only one point (either from $L$ or $R$).

# Data structures

**Initialization:** sort the points, with respect to $x$.
Cost: $\boldsymbol{O(n\log(n))}$, before recursive calls
**Note:** Merge based on the $y$-coordinate so that the result is sorted in $y$-coordinate.

**Then:** recursion

- finding the $x$-median is easy $\boldsymbol{O(1)}$
- for the next recursive calls, split the sorted lists $\boldsymbol{O(n)}$
- remove the points at distance $\geq \delta$ from the $x$-splitting line $\boldsymbol{O(n)}$
- inspect all remaining points in increasing $y$-order. For each of them, compute the distance to the next 8 points and keep the min. $\boldsymbol{O(n)}$

**Runtime:** $\boldsymbol{T(n) = 2T(n/2) + \Theta(n)}$ so $\boldsymbol{T(n) \in \Theta(n\log(n))}$

# Beyond the master theorem: median of medians

**Median:** given $A[0..n-1]$, find the entry that would be at index $\lfloor n/2 \rfloor$ if $A$ was sorted

**Selection:** given $A[0..n-1]$ and $k$ in $\{0, \ldots, n-1\}$, find the entry that would be at index $k$ if $A$ was sorted **Known results:**

sorting $A$ in $\boldsymbol{O(n \log(n))}$, or a simple randomized algorithm in expected time $\boldsymbol{O(n)}$

# The selection algorithm

**quick-select**$(A, k)$
$A$: array of size $n$, $k$: integer s.t. $0 \le k < n$
1. $p \leftarrow$ **choose-pivot**$(A)$
2. $i \leftarrow$ **partition**$(A, p)$ — $i$ is the correct index of $p$
3. **if** $i = k$ **then**
4. **return** $A[i]$
5. **else if** $i > k$ **then**
6. **return quick-select**$(A[0, 1, \ldots, i-1], k)$
7. **else if** $i < k$ **then**
8. **return quick-select**$(A[i+1, i+2, \ldots, n-1], k-i-1)$

**Question:** how to find a pivot such that both $\boldsymbol{i}$ and $\boldsymbol{n-i-1}$ are not too large?

# Median of medians

**Sketch of the algorithm:**

- divide $A$ into $n/5$ groups $G_1, \ldots, G_{n/5}$ of size 5
- find the medians $m_1, \ldots, m_{n/5}$ of each group $\boldsymbol{O(n)}$
- pivot $p$ is the median of $[m_1, \ldots, m_{n/5}]$ $\boldsymbol{T(n/5)}$

**Claim**

With this choice of $p$, the indices $\boldsymbol{i}$ and $\boldsymbol{n-i-1}$ are at most $\boldsymbol{7n/10}$

**Proof**

- **half** of the $m_i$'s are greater than $p$ $\boldsymbol{n/10}$
- for each $m_i$, there are **3** elements in $G_i$ greater than or equal to $m_i$
- so **at least $\boldsymbol{3n/10}$** elements greater than $p$
- so **at most $\boldsymbol{7n/10}$** elements less than $p$
- so $i$ is **at most $\boldsymbol{7n/10}$**. Same thing for $n - i - 1$

**Consequence:** the runtime $T(n)$ satisfies

$$T(n) \leq T(n/5) + T(7n/10) + O(n)$$

**Claim**

This gives $\boldsymbol{T(n) \in O(n)}$