10.10
Supplemental: Divide and conquer for closest pair
Problem: Closest pair

\( P \): Set of \( n \) distinct points in the plane.
Compute the two points \( p, q \in P \) that are closest together. Formally, compute

\[
\arg \min_{p,q \in P : p \neq q} \|p - q\|.
\]
Closest pair: Divide and conquer leads to a special case

\[ P = P_L \cup P_R \]
Closest pair: Divide and conquer leads to a special case

\[ P = P_L \cup P_R \]
Closest pair: Divide and conquer leads to a special case

$P = P_L \cup P_R$
Closed pair: Divide and conquer leads to a special case

\[ P = P_L \cup P_R \]
Closest pair: Divide and conquer leads to a special case

1. \( P = P_L \cup P_R \)
2. \( |P_L| = |P_R| = \frac{n}{2} \).
   \( x(P_L) < 0 \) and \( x(P_R) > 0 \).
Closest pair: Divide and conquer leads to a special case

1. $P = P_L \cup P_R$
2. $|P_L| = |P_R| = n/2$.
   $x(P_L) < 0$ and $x(P_R) > 0$.
3. Given $\ell = \min(cp(P_L), cp(P_R))$. 

\[\text{Diagram with points on a plane divided by a vertical line.}\]
Closest pair: Divide and conquer leads to a special case

1. \( P = P_L \cup P_R \)
2. \(|P_L| = |P_R| = n/2.\)

   \( x(P_L) < 0 \) and \( x(P_R) > 0.\)
3. Given \( \ell = \min(cp(P_L), cp(P_R)) \).
4. \( P_m = \{p \in P \mid -\ell \leq x(p) \leq \ell\} \)
Closest pair: Divide and conquer leads to a special case

1. \( P = P_L \cup P_R \)
2. \(|P_L| = |P_R| = n/2.\)
   \( x(P_L) < 0 \) and \( x(P_R) > 0. \)
3. Given \( \ell = \min(cp(P_L), cp(P_R)). \)
4. \( P_m = \{ p \in P \mid -\ell \leq x(p) \leq \ell \} \)
5. Task: compute \( cp(P) = \min(\ell, cp(p_M)). \)
Closest pair: Divide and conquer leads to a special case

1. \( P = P_L \cup P_R \)

2. \(|P_L| = |P_R| = n/2.\)
   \(x(P_L) < 0\) and \(x(P_R) > 0.\)

3. Given \( \ell = \min(cp(P_L), cp(P_R)).\)

4. \( P_m = \{ p \in P \mid -\ell \leq x(p) \leq \ell \} \)

5. Task: compute \( cp(P) = \min(\ell, cp(p_M)) \).

6. **Claim:** Closest pair in \( P_m \) can be computed in \( O(n \log n) \) time.
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator cannot be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full

...or $P_m$ is well spread
An elevator can not be too full
...or $P_m$ is well spread
An elevator can not be too full

...or \( P_m \) is well spread

Closet pair in \( P_m \) can be computed in 
\( O(n \log n) \) time.
An elevator can not be too full

...or $P_m$ is well spread

Closet pair in $P_m$ can be computed in $O(n \log n)$ time.

Closet pair in $P_m$ can be computed in $O(n)$ time, if $P$ is presorted by $y$-order.
Closest pair: Algorithm

\[ \text{CPDInner} = \text{ClosestPairDistance} \]

```plaintext
CPDInner( P = \{p_1, \ldots, p_n\} ):
    if \(|P| = O(1)\) then compute by brute force
    \(x^* = \text{median}(x(p_1), \ldots, x(p_n))\).
    \(P_L \leftarrow \{p \in P \mid x(p) \leq x^*\}\)
    \(P_R \leftarrow \{p \in P \mid x(p) > x^*\}\)
    \(\ell_L = \text{CPDInner}(P_L)\)
    \(\ell_R = \text{CPDInner}(P_R)\)
    \(\ell = \min(\ell_L, \ell_R)\).
    \(P_m = \{p \in P \mid x^* - \ell \leq x(p) \leq x^* + \ell\}\)
    \(\ell_M = \text{call alg. closest-pair distance for special case on } P_m\).
    return \(\min(\ell, \ell_M)\).
```

\[ \text{CPD}( P = \{p_1, \ldots, p_n\} ):\]
    return \(\text{CPDInner}(P)\)
Given a set $P$ of $n$ points in the plane, one can compute the closest pair distance in $P$ in $O(n \log^2 n)$ time.
Closest pair: Algorithm

**CPDInner = ClosestPairDistance**

```c
CPDInner( P = \{p_1, \ldots, p_n\} ):
    if |P| = O(1) then compute by brute force
    x^* = \text{median}(x(p_1), \ldots, x(p_n)).
    P_L \leftarrow \{p \in P | x(p) \leq x^*\}
    P_R \leftarrow \{p \in P | x(p) > x^*\}
    \ell_L = \text{CPDInner}(P_L)
    \ell_R = \text{CPDInner}(P_R)
    \ell = \min(\ell_L, \ell_R).
    P_m = \{p \in P | x^* - \ell \leq x(p) \leq x^* + \ell\}
    \ell_M = \text{call alg. closest-pair distance for special case on } P_m.
    return \min(\ell, \ell_M).
```

**CPD( P = \{p_1, \ldots, p_n\} ):**

Sort $P$ by $x$-order. Sort $P$ by $y$-order

return $\text{CPDInner}(P)$
Theorem

Given a set $P$ of $n$ points in the plane, one can compute the closest pair distance in $P$ in $O(n \log n)$ time.
Wait wait... one can do better

Rabin showed that if we allow the floor function, and randomization, one can do better:

**Theorem**

*Given a set $P$ of $n$ points in the plane, one can compute the closest pair distance in $P$ in $O(n)$ time.*
THE END

... (for now)