5.4 Closest Pair of Points
Closest Pair of Points

Closest pair. Given n points in the plane, find a pair with smallest Euclidean distance between them.

Fundamental geometric primitive.
- Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.
- Special case of nearest neighbor, Euclidean MST, Voronoi.
  \ fast closest pair inspired fast algorithms for these problems

Brute force. Check all pairs of points p and q with $\Theta(n^2)$ comparisons.

1-D version. $O(n \log n)$ easy if points are on a line.

Assumption. No two points have same x coordinate.

| to make presentation cleaner |
Closest Pair of Points: First Attempt

**Divide.** Sub-divide region into 4 quadrants.
Closest Pair of Points: First Attempt

Divide. Sub-divide region into 4 quadrants.
Obstacle. Impossible to ensure $n/4$ points in each piece.
Closest Pair of Points

Algorithm.

- **Divide:** draw vertical line \( L \) so that roughly \( \frac{1}{2} n \) points on each side.
Closest Pair of Points

**Algorithm.**
- **Divide:** draw vertical line $L$ so that roughly $\frac{1}{2} n$ points on each side.
- **Conquer:** find closest pair in each side recursively.
Closest Pair of Points

Algorithm.
- **Divide:** draw vertical line $L$ so that roughly $\frac{1}{2}n$ points on each side.
- **Conquer:** find closest pair in each side recursively.
- **Combine:** find closest pair with one point in each side. — seems like $\Theta(n^2)$
- Return best of 3 solutions.
Closest Pair of Points

Find closest pair with one point in each side, assuming that distance < δ.
Find closest pair with one point in each side, *assuming that distance* $< \delta$.

Observation: only need to consider points within $\delta$ of line $L$. 

$\delta = \min(12, 21)$
Closest Pair of Points

Find closest pair with one point in each side, assuming that distance < $\delta$.

Observation: only need to consider points within $\delta$ of line $L$.

Sort points in $2\delta$-strip by their $y$ coordinate.
Closest Pair of Points

Find closest pair with one point in each side, assuming that distance $< \delta$.

- Observation: only need to consider points within $\delta$ of line $L$.
- Sort points in $2\delta$-strip by their $y$ coordinate.
- Only check distances of those within 11 positions in sorted list!

\[ \delta = \min(12, 21) \]
Closest Pair of Points

**Def.** Let $s_i$ be the point in the $2\delta$-strip, with the $i^{th}$ smallest $y$-coordinate.

**Claim.** If $|i - j| \geq 12$, then the distance between $s_i$ and $s_j$ is at least $\delta$.

**Pf.**

- No two points lie in same $\frac{1}{2}\delta$-by-$\frac{1}{2}\delta$ box.
- Two points at least 2 rows apart have distance $\geq 2(\frac{1}{2}\delta)$. □

**Fact.** Still true if we replace 12 with 7.
Closest-Pair Algorithm

Closest-Pair(p₁, ..., pₙ) {
    Compute separation line L such that half the points are on one side and half on the other side.

    δ₁ = Closest-Pair(left half)
    δ₂ = Closest-Pair(right half)
    δ = min(δ₁, δ₂)

    Delete all points further than δ from separation line L

    Sort remaining points by y-coordinate.

    Scan points in y-order and compare distance between each point and next 11 neighbors. If any of these distances is less than δ, update δ.

    return δ.
}
Running time.

\[ T(n) \leq 2T(n/2) + O(n \log n) \implies T(n) = O(n \log^2 n) \]

**Q.** Can we achieve \( O(n \log n) \)?

**A.** Yes. Don’t sort points in strip from scratch each time.
- Each recursive returns two lists: all points sorted by \( y \) coordinate, and all points sorted by \( x \) coordinate.
- Sort by **merging** two pre-sorted lists.

\[ T(n) \leq 2T(n/2) + O(n) \implies T(n) = O(n \log n) \]
5.5 Integer Multiplication
**Integer Arithmetic**

**Add.** Given two n-digit integers $a$ and $b$, compute $a + b$.
- $O(n)$ bit operations.

**Multiply.** Given two n-digit integers $a$ and $b$, compute $a \times b$.
- Brute force solution: $\Theta(n^2)$ bit operations.
Divide-and-Conquer Multiplication: Warmup

To multiply two $n$-digit integers:

- Multiply four $\frac{1}{2}n$-digit integers.
- Add two $\frac{1}{2}n$-digit integers, and shift to obtain result.

$x = 2^{n/2} \cdot x_1 + x_0$

$y = 2^{n/2} \cdot y_1 + y_0$

$xy = \left(2^{n/2} \cdot x_1 + x_0\right) \left(2^{n/2} \cdot y_1 + y_0\right) = 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0$

$T(n) = 4T(n/2) + \Theta(n)$

assuming $n$ is a power of 2
Karatsuba Multiplication

To multiply two n-digit integers:

- Add two $\frac{1}{2}n$ digit integers.
- Multiply three $\frac{1}{2}n$-digit integers.
- Add, subtract, and shift $\frac{1}{2}n$-digit integers to obtain result.

**Theorem.** [Karatsuba-Ofman, 1962] Can multiply two n-digit integers in $O(n^{1.585})$ bit operations.

\[
\begin{align*}
x &= 2^{n/2} \cdot x_1 + x_0 \\
y &= 2^{n/2} \cdot y_1 + y_0 \\
xy &= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0 \\
&= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0 + x_0 y_0
\end{align*}
\]

**Theorem.** [Karatsuba-Ofman, 1962] Can multiply two n-digit integers in $O(n^{1.585})$ bit operations.

\[
\begin{align*}
T(n) &\leq T\left(\left\lfloor n/2 \right\rfloor \right) + T\left(\left\lceil n/2 \right\rceil \right) + T\left(1 + \left\lceil n/2 \right\rceil \right) + \Theta(n) \\
\Rightarrow T(n) &= O(n^{\log_2 3}) = O(n^{1.585})
\end{align*}
\]
Karatsuba: Recursion Tree

\[ T(n) = \begin{cases} 
0 & \text{if } n = 1 \\
3T(n/2) + n & \text{otherwise} 
\end{cases} \]

\[ T(n) = \sum_{k=0}^{\log_2 n} n \left(\frac{3}{2}\right)^k = \frac{(\frac{3}{2})^{1+\log_2 n} - 1}{\frac{3}{2} - 1} = 3n^{\log_2 3} - 2 \]