Closest Pair

$n$ points in 2-D plane. Which 2 points are closest together?

\[ d_{\text{min}} = \min_{i \neq j} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \]

Assume all $x_i$ are distinct. Also, all $y_i$ are distinct.

For simplicity, $n$ is a power of 2.

Brute force solution:

\[ d_{\text{min}} = \infty \]

\[ \text{for } (i = 1; i < n; i++) : \]

\[ f(j = i+1; j < n; j++) : \]

\[ d = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \]

\[ \text{if } d < d_{\text{min}} : \]

\[ d_{\text{min}} = d \]

Runtime? $O(n^2)$
Closest Pair (Divide-and-Conquer)

Possible subproblems?
How are they made?
How are solutions recombined?
What defines the size of the problem?
Closest Pair (Divide-and-conquer)

Divide the problem into 2 equal parts, L and R.
L = \{ half of the points with lowest x values \}
R = \{ half of the points with highest x values \}

Find the closest pair in L, and in R. Distances are \( d_L, d_R \).
\( d = \min(d_L, d_R) \)

Find closest pair between L and R, if it is \( < d \).
Solution is smallest distance.

what is the base case?
How do you find closest pair between L and R?

Only consider points within \( l \) of division. Why?
Sort them by \( y \). Compare each point with 7 closest points in \( y \).
why 7?
def closest_pair(p[1:n]):
    if len(p) <= 1: return ∞
    sort p by x
    d_L = closest_pair(p[1:n/2])
    d_R = closest_pair(p[n/2+1:n])
    d = min(d_L, d_R)

    X_mid = (p[n/2].x + p[n/2+1].x) / 2
    q = all points in p with |p_i - x_mid| ≤ d
    sort q by y

    for (i=1; i ≤ |q| - 7; i++):
        for (j=i+1; j ≤ i+7; j++):
            t = dist(q_i, q_j)
            if t < d:
                d = t

Recurrence relation?
Subproblems? Count? Size?
Divide complexity?
Combine complexity?
Closest Pair  (Divide-and-Conquer)

\[ T(n) = 2T\left(\frac{n}{2}\right) + O(n \log n) \]

Solve by substitution:

\[ T(n) = 2T\left(\frac{n}{2}\right) + O(n \log n) \quad k = 0 \]
\[ T(n/2) = 2T\left(\frac{n}{4}\right) + \frac{n}{2} \log \frac{n}{2} \]
\[ T(n) = 2 \left[ 2T\left(\frac{n}{4}\right) + \frac{n}{2} \log \frac{n}{2} \right] + n \log n \]
\[ T(n/4) = 2T\left(\frac{n}{8}\right) + \frac{n}{2^2} \log \frac{n}{2^2} \]
\[ T(n) = 2^2 \left[ 2T\left(\frac{n}{8}\right) + \frac{n}{2^2} \log \frac{n}{2^2} \right] + 2 \frac{n}{2} \log \frac{n}{2} + n \log n \]
\[ T(n/8) = 2^3 T\left(\frac{n}{16}\right) + \frac{n}{2^3} \log \frac{n}{2^3} + 2 \frac{n}{2^3} \log \frac{n}{2^3} + n \log n \]
\[ T(n) = 2^3 \left[ 2T\left(\frac{n}{16}\right) + \frac{n}{2^3} \log \frac{n}{2^3} \right] + \frac{n}{2^3} \log \frac{n}{2^3} + \frac{n}{2^3} \log \frac{n}{2^3} + n \log n \]

\[ T(n) = 2^k T\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k} n \log(n/2^i) \]

\[ \frac{n}{2^{k+1}} = 1 \Rightarrow n = 2^{k+1} \Rightarrow \log_2 n = k+1 \Rightarrow k_{\text{max}} = \log_2 n - 1 \]

\[ T(n) = 2^{k_{\text{max}}+1} T(1) + \sum_{i=0}^{k_{\text{max}}} n \log(n/2^i) \]
\[ T(n) = \sum_{i=0}^{\log_2 n - 1} n \log(n/2^i) = \sum_{i=0}^{\log_2 n - 1} (n \log n - n \log(2^i)) = O(n(\log n)^2) - O(n \log n) \]
\[ = O(n (\log n)^2) \]