\[ T(n) \leq n + \frac{1}{2} T\left(\frac{3n}{4}\right) + \frac{1}{2} T(n) \]

Multiply both sides by 2:

\[ 2T(n) \leq 2n + T\left(\frac{3n}{4}\right) + T(n) \]

\[ T(n) \leq 2n + T\left(\frac{3n}{4}\right) \]

Master's: \( a = 1 \), \( b = \frac{4}{3} \), \( i = 1 \)

\[ a < b^i \quad \text{Root dominates} \quad \Rightarrow \Theta(n^i) \]

If our equation was \( T(n) = \) \( \Theta \)
then we could say \( T(n) = \Theta(n^i) \) but since our equation is \( T(n) \leq \) we can say:

\[ T(n) = O(n) \]

Important Note:
We showed average case is \( O(n) \)
Can solve in worst-case \( O(n) \) (Ch 9.3)
using Median-of-Medians to find pivot:
- Better worst-case time but
- Much more complicated
- Almost always slower in practice
- Has \( O(n) \) time but \( c \) is high!
Closest Points Problem

Input: list P of n points on Euclidean plane

\[ P_1 = (x_1, y_1), P_2 = (x_2, y_2), \ldots, P_n = (x_n, y_n) \]

Goal: Find a closest pair of points

Applications: Air/sea traffic controller use to detect closest vehicles to avoid collisions

For simplicity, we will find mind distance (instead of points)

Distance blw 2 points \( P_i, P_j \):

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

Brute force: \( n \) points, so \( n(n-1)/2 \) pairs,

Compute distances blw all pairs, find minimum.
\[ \Rightarrow O(n^2) \]

How to do better?

Intuitively

Probably shouldn't check these points
How to account for this?
- Partition the points: left, right.

Now what? Same problem within $P_L, P_R$:

So, continuously (recursively) split each subspace in half.

When to stop recursing? When subspace contains exactly $\frac{7}{3}$ points.

Now, what?
- Find distance b/w pair in $P_L$
  "" "" "" $P_R$
- "" "" "" all pairs crossing partition
- Find min distance of these
Preprocess (P)
Sort P by x-coord.

1. Closest (P) // P ∈ 1..nJ : list of n points
2. // Finds distance bw closest pair in P
3. if (|P| = \( \leq 3 \)) return dist(P_1,P_2) //using distance formula
4. else
   \( \leq \) min distance of all pairs
5. \( P_L, P_R = \) Partition (P) // partition P into left, right
6. \( d_L = \) Closest (P_L) // closest distance on left
7. \( d_R = \) Closest (P_R) // " " " right
   \( d = \min(d_L,d_R) \)
8. \( d_c = \) closestCrossing (P_L,P_R,d) // closest dist amng crossing
   <What to eventually return?> // pairs.
9. return min(d_L,d_R,d_c)

<Overview of Algorithm>

Questions:
(1) How to partition?
(2) What if subspace contains only 1 point?
(3) How to find closest distance among crossing pairs in < O(n^2) time?
1) Partition: "How to partition into $P_L, P_R$?"

Partition $(P)$
- sort $P$ by $x$-coord

$P_L = P[1 \ldots \lfloor n/2 \rfloor]$
$P_R = P[\lceil n/2 + 1 \rceil \ldots n]$

return $P_L, P_R$.

Problem? Partition() called for every recursive call
So sort required each time.

Solution: Sort $P$ once at the beginning.

$\langle$ Preprocess $(P)$ $\rangle$ ← add to Closest() code,
$\langle$ Sort $P$ by $x$-coord $\rangle$ remove from Partition() (2) "What if subset contains only 1 point?"