Divide-and-Conquer Algorithms

1. **Divide** - divide problem into subproblems (smaller input for same problem)

2. **Conquer** - recursively solve subproblems (or directly solve when input reaches a termination condition)

3. **Combine** - use subproblem solutions to solve given problem
Mergesort \( a[0], \ldots, a[n-1] \)

1. Divide - split array in half
   - input (problem) sort \( a[p] \ldots a[r] \)
   - subproblem \( q = \lfloor (p+r)/2 \rfloor \)
   - sort \( a[p] \ldots a[q] \) + \( a[q+1] \ldots a[r] \)

2. Recursively sort two subarrays

3. Merge two sorted subarrays
Example Execution

6 4 3 8 1 7 2 5

3 4 6 8 1 2 5 7

temp

array

This takes linear time (in size of resulting subarray)
Closest pair of points

Divide, split into left half and a right half of \( \sim n/2 \) points each

Recursively find closest pair on each of these two subproblems

Combine - use subproblem sols to find closest pair in given point set
For \( i = 0; i < \text{numInStrip} - 1; i++ \) do

\[
\text{For } j = i + 1 ; \\
\text{While } ( j < \text{numInStrip} \&\& \text{yStrip}(j).y - \text{yStrip}(i).y < d ) \\
\quad \quad d = \min(d, \text{distance point } i + \text{ point } j) \\
\quad \quad j++ \\
\text{3}
\]

consider each point in yStrip looking upward
Closest pair of points (cont.)

Combine step in more detail:

\[ d = \min(d_L, d_R) \]

**Step 1**

- Construct y-Strip to ref all points in order of pts By Y (sorted by y coord) with x coord > \( x_{R-d} \)
- \( x < x_{L+d} \)

**Step 2**

- Find any pair with y-Strip with distance < \( \Delta \)
  - (one from each side)
Example Execution

pts by X

\[ d = \min(d_L, d_R) \]
do it.

right point from any
with

right half

left subproblem

right subproblem

no point here is a closest pair with left half

$x-x_d$
Correctness

1. If closest pair of points are both on left we find in recursive call for left. Same for right.

2a) We've argued that any pair of points not in y-strap need only be considered by 1.

2b) We've argued that any pair in y-strap that could have dist < d is considered.
Correctness (cont.)
$T(n) = \text{time complexity for n points}$

Analysis of Asymptotic Time Complexity

**Divide** can be done in linear time. You must create $\text{pts By X Left}$, $\text{pts By X Right}$, $\text{pts By Y Left}$, $\text{pts By Y Right}$ carefully.

**Recursive calls**

$2T(n/2)$

**Combine**—argue that this is linear.
Time complexity of Combine step \( \leq 2d \)

Consider at most 5 points in "while" loop for "point i".
Expressing Time Complexity using Recurrence Equation

\[ T(n) = c_1 \cdot n + 2T\left(\frac{n}{2}\right) + c_2 \cdot n \]

\[ = 2T\left(\frac{n}{2}\right) + c_1 n \]

\[ = c_1 + c_2 \]