EE 360C ALGORITHMS - FALL 2018 - October 25 - 11-12:30

Example 1: $T(n) = 3\,T(n/4) \cdot n\log n$

Applying M.T. to this recursion:

- $a = 3$, $b = 4$, $f(n) = n\log n$

- $f(n) = n\log n$ vs $n^{\log_b a} = n^{0.793\ldots}$

  $\rightarrow f(n) > n^{\log_b a}$, therefore we are in Case 3.

  (1) For $\varepsilon = 0.2$: $f(n) = \Omega(n^{\log_b a + \varepsilon})$ ✓

  (2) We have to check if also $a f(n/b) \le c \cdot f(n)$ for some $c < 1$.

  $3 f(n/4) = 3 \frac{n}{4} \log \frac{n}{4} \le c \cdot n\log n$ ✓

  $\Rightarrow T(n) = \Theta(f(n)) = \Theta(n\log n)$.

Example 2: $T(n) = 2\,T(n/2) + n\log n$

- $f(n) = n\log n$ vs $n^{\log_b a} = n^{\log_2 2} = n$

  $\rightarrow f(n)$ is larger, but not polynomially larger.

  (For it to be polynomially larger, we would need something of the form $n^{1+\varepsilon} = n \cdot n^{\varepsilon}$).

  Therefore this does not belong to any of the three cases.

DIVIDE AND CONQUER: CLOSEST PAIR OF POINTS

Given n points, find the pair with the smallest distance between them.

Computational Geometry

A naive algorithm would need $O(n^2)$ time.

1D CASE

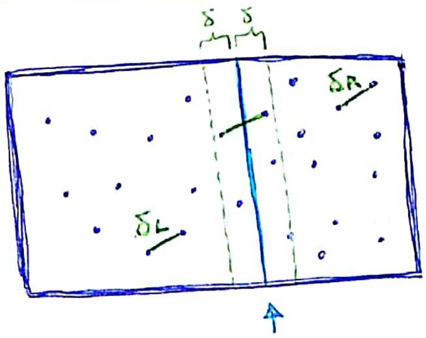
In a situation where the points are in one line, we can create an $O(n\log n)$ divide and conquer algorithm:

- Sort
- Compute distances between consecutive points, remembering the smallest distance.

This works because closest point is next to you in the order.

2D CASE

where $\delta = \min(\delta_L, \delta_R)$

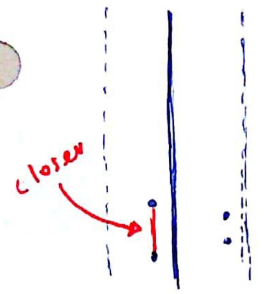


dividing line: $\sim \frac{n}{2}$ points on each side

- Divide the space in two parts, each containing $\sim \frac{n}{2}$ points.
- Find the closest pair in each side recursively. (δ_L, δ_R)
- ★ Find the closest pair with one point on each side
- Return the best of the 3 solution

★ Only need to consider the set of points S within $\delta = \min(\delta_L, \delta_R)$ of the dividing line.

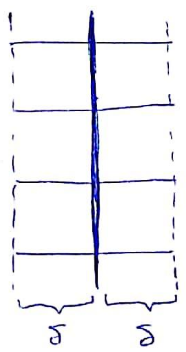
Lets sort S by y-coordinate to get S_y, and compute distances between consequtive points



Instead compute distances to points within $\boxed{8}$ positions in S_y.

Claim: If two points are more than 8 positions apart in S_y (S sorted by y-coordinate) then the distance between them is $> \delta$.

Pf: Visualize $\frac{1}{2}\delta \times \frac{1}{2}\delta$ boxes near the dividing line.



Points in S_y must lie in these boxes.

Key Observation: There is at most 1 point in every box

>8 positions apart $\Rightarrow$ >2 rows apart
$\Rightarrow$ $>\delta$ apart. □

Sort points by x (P_x) and, independently by their y coordinates (P_y)

Closest Pair(P_x, P_y):

If only 2 points, return their distance.
δ_L = ClosestPair(lefthalf(P_x, P_y)) $T(n/2)$
δ_R = ClosestPair(righthalf,(P_x, P_y)) $T(n/2)$
$\delta = \min(\delta_L, \delta_R)$
S_y = Points in R_y within δ of dividing line
for $i=1$ to $|S_y|$
  for $j=i+1$ to $i+1$
    $\delta := \min(d(S_y[i], S_y[j]), \delta)$
Return δ.

$\left.\right\} O(n)$

$T(n) = 2T(n/2) + O(n)$
$\Downarrow$
$O(n \log n)$