S. KIMMEL

Algorithm

Preprocessing step:

Create arrays X = points sorted by x-coordinate, Y = points sorted by y-coordinate $\Big\}\ O(n\log n)$

Closest-Pair(X, Y)

- If $(|X| \le 3)$ { check all possible pairs & return min distance } $O(1)$

- Create X_L, Y_L = sorted left half by x, y
- Create X_R, Y_R = " right " $\Big\}\ O(n)$
- $\delta = \min\{\text{Closest-Pair}(X_R, Y_R), \text{Closest-Pair}(X_L, Y_L)\} \leftarrow 2 \cdot T\left(\frac{n}{2}\right)$
- Y_δ = points within δ of line sorted by y $O(n)$
- for $(i = 1$ to length$(Y_\delta) - 7)$ {
  - for $(j = 1$ to $7)$ {
    - if $(d(p_i, p_{i+j}) < \delta)$ {
      - $\delta = d(p_i, p_{i+j})$
    - }
  - }
- } $\Big\}\ \sim 11n = O(n)$
- return δ

(only returns shortest distance, but can easily modify to return closest points)

Loop over points in Y_δ, and check distance between current and next 7 pts. Track smallest distance

S.KIMMEL

Recurrence Relation:

problem

$T(n)$ = max # of operations required on instance with n points

<u>Base</u>: $T(3), T(2) <$ constant

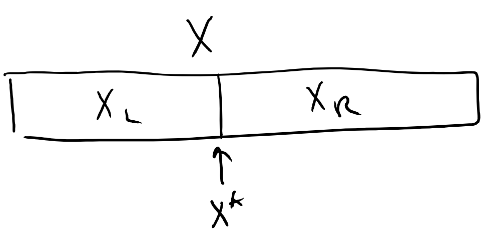
<u>Recurrence</u>: $n > 3$

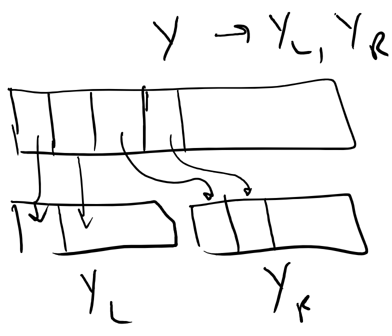
A) $T(n) \le 2T\left(\frac{n}{2}\right) + O(n)$

B) $T(n) \le 2T\left(\frac{n}{2}\right) + O(n^2)$

C) $T(n) \le \frac{1}{2}T(2n) + O(n)$

D) $T(n) \le \frac{1}{2}T(2n) + O(n^2)$





Stay in sorted order! Takes Time $O(n)$

With group, go over algorithm analysis: take turns explaining.