Closest Pair

Piotr Indyk
Closest Pair

• Find a closest pair among $p_1 \ldots p_n \in \mathbb{R}^d$

• Easy to do in $O(dn^2)$ time
  – For all $p_i \neq p_j$, compute $||p_i - p_j||$ and choose the minimum

• We will aim for better time, as long as $d$ is “small”

• For now, focus on $d=2$
Divide and conquer

• Divide:
  – Compute the median of x-coordinates
  – Split the points into $P_L$ and $P_R$, each of size $n/2$

• Conquer: compute the closest pairs for $P_L$ and $P_R$

• Combine the results (the hard part)
Combine

- Let $k = \min(k_1, k_2)$
- Observe:
  - Need to check only pairs which cross the dividing line
  - Only interested in pairs within distance $< k$
- Suffices to look at points in the $2k$-width strip around the median line
Scanning the strip

- Sort all points in the strip by their y-coordinates, forming $q_1 ... q_t$, $t \leq n$.
- Let $y_i$ be the y-coordinate of $q_i$
- $k_{\text{min}} = k$
- For $i = 1$ to $t$
  - $j = i - 1$
  - While $y_i - y_j < k$
    - If $||q_i - q_j|| < k_{\text{min}}$ then $k_{\text{min}} = ||q_i - q_j||$
    - $j := j - 1$
- Report $k_{\text{min}}$ (and the corresponding pair)
Analysis

• Correctness: easy
• Running time is more involved
• Can we have many \( q_i \)'s that are within distance \( k \) from \( q_i \)?
• No
• Proof by \textit{packing} argument
Analysis, ctd.

**Theorem:** there are at most 7 $q_j$’s, $j<i$, such that $y_i - y_j \leq k$.

**Proof:**

- Each such $q_i$ must lie either in the left or in the right $k \times k$ square.
- Within each square, all points have distance distance $\geq k$ from others.
- We can pack at most 4 such points into one square, so we have 8 points total (incl. $q_i$).
At most 4

• Split the square into 4 sub-squares of size \( \frac{k}{2} \times \frac{k}{2} \)
• Diameter of each square is \( \frac{k}{2^{1/2}} < k \) → at most one point per sub-square
Running time

• Divide: $O(n)$
• Combine: $O(n \log n)$ because we sort by $y$
• However, we can:
  – Sort all points by $y$ at the beginning
  – Divide preserves the $y$-order of points
  Then combine takes only $O(n)$
• We get $T(n)=2T(n/2)+O(n)$, so $T(n)=O(n \log n)$
Higher dimensions (sketch)

• Divide: split $P$ into $P_L$ and $P_R$ using the hyperplane $x=t$

• Conquer: as before

• Combine:
  – Need to take care of points with $x$ in $[t-k, t+k]$
  – This is essentially the same problem, but in $d-1$ dimensions
  – We get:
    • $T(n,d) = 2T(n/2, d) + T(n,d-1)$
    • $T(n,1) = O_d(1) n$
  – Solves to: $T(n,d) = n \log^{d-1} n$
Closest Pair with Help

- Given: $P=\{p_1, \ldots, p_n\}$ of points from $\mathbb{R}^d$, such that the closest distance is in $(t, c \cdot t]$.
- Goal: find the closest pair.
- Will give an $O((2c \cdot d^{1/2})^d \cdot n)$ time algorithm.
- Note: by scaling we can assume $t=1$. 
Algorithm

- Impose a cubic grid onto $\mathbb{R}^d$, where each cell is a $\frac{1}{d^{1/2}} \times \frac{1}{d^{1/2}}$ cube
- Put each point into a bucket corresponding to the cell it belongs to
- Diameter of each cell is $\leq 1$, so at most one point per cell
- For each $p \in P$, check all points in cells intersecting a ball $B(p, c)$
- How many cells are there?
  - All are contained in a $d$-dimensional box of side $2(c + 1/d^{1/2}) \leq 2(c + 1)$
  - At most $(2d^{1/2}(c+1))^d$ such cells
- Total time: $O((2c d^{1/2})^d n)$
How to find good $t$?

- Repeat:
  - Choose a random point $p$ in $P$
  - Let $t = t(p) = D(p, P\setminus\{p\})$
  - Impose a grid with side $t' < t/(1+d^{1/2})$, i.e., such that any pair of adjacent cells has diameter $<t$
  - Put the points into the grid cells
  - Remove all points whose all adjacent cells are empty

- Until $P$ is empty
- Observation: the values $t$ are decreasing
Correctness

• Consider $t$ computed in the last iteration
  – There is a pair of points with distance $t$ (it defines $t$)
  – There is no pair of points with distance $t'$ or less
    (otherwise they would have been placed in adjacent cells, and the algorithm would have continued)
  – We get $c = \frac{t}{t'} \sim 2 d^{1/2}$
Running time

- Consider $t(p_1) \ldots t(p_m)$
- An iteration is lucky if $t(p_i) \geq t$ for at least half of points $p_i$
- The probability of being lucky is $\geq 1/2$
- Expected number of iterations till a lucky one is $\leq 2$
- After we are lucky, the number of points is $\leq m/2$
- Total expected time = $3^d$ times $O(n+n/2+n/4+\ldots+1)$