CS 341 – Algorithms

Lecture 4 – Divide and Conquer II

21 May 2021

Today’s Plan

1. Closest Pair

2. Arithmetic Problems

1. Homework 1 posted

2. Supplementary exercises

3. First tutorial on Tuesday

Closest Pair

Input: n points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ on the 2-D plane.

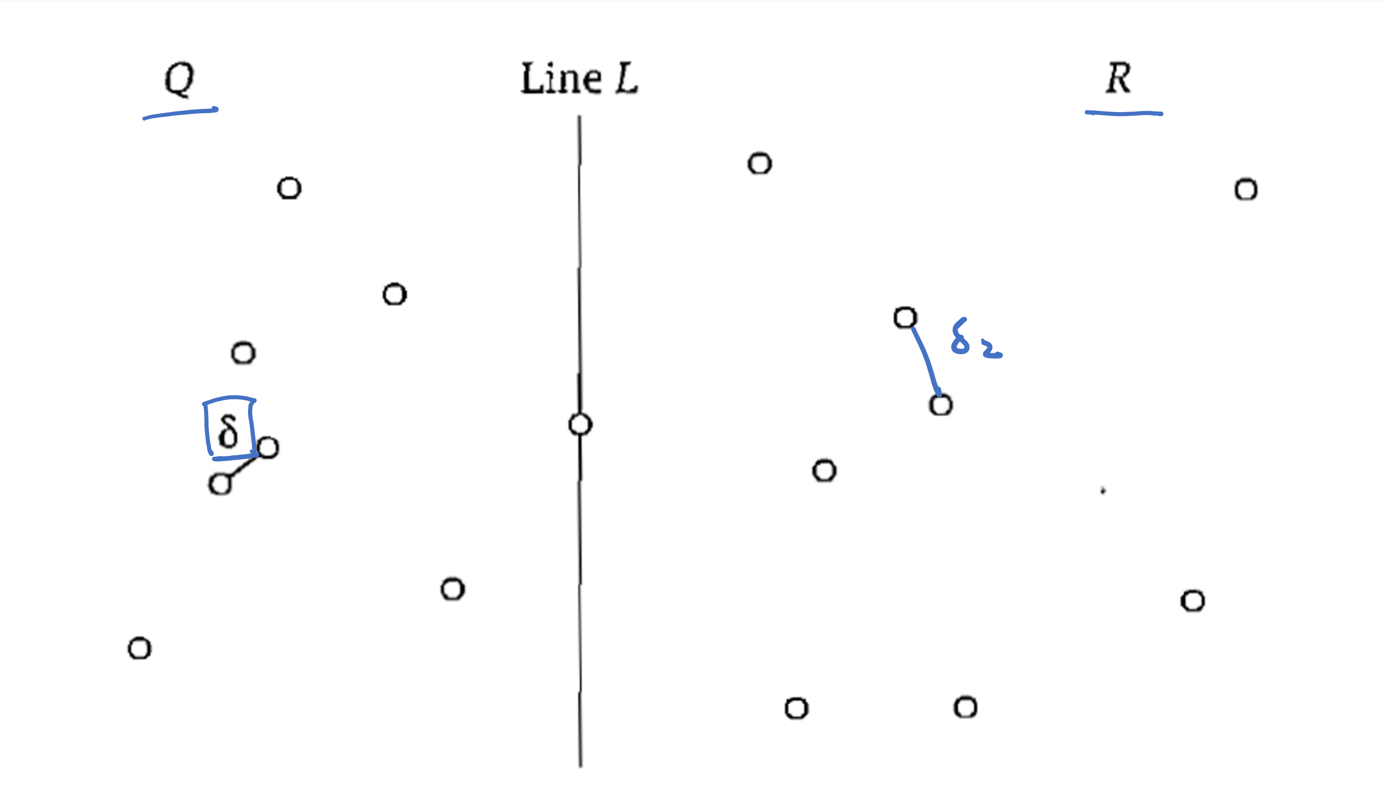
Output: a pair $1 \leq i < j \leq n$ that minimizes $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Assumption: all the x-values are distinct.

naive: $O(n^2)$

Divide and Conquer

Divide into two halves. Find a closest pair within Q. Find a closest pair within R.



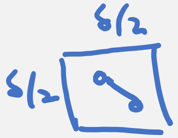
Let δ be the minimum distance that we have found so far.

Crossing Pair with Distance $< \delta$

It remains to find out whether there is a pair with $(x_i, y_i) \in Q$ and $(x_j, y_j) \in R$ with $\boxed{\text{distance} < \delta}$.

Obs 1 : Each box has at most one point.

proof

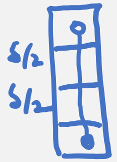


distance $\leq \frac{\delta}{\sqrt{2}} < \delta$

Contradict that closest pair in Q, R has distance δ. □

Obs 2 Points that are two "layers" apart have distance > δ.

proof



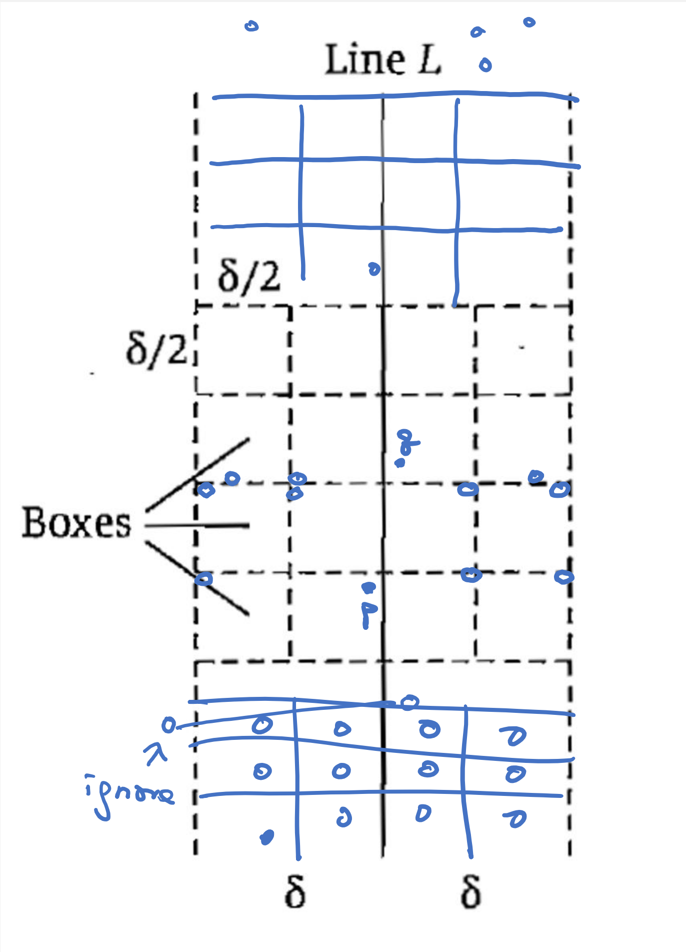
distance > δ

□

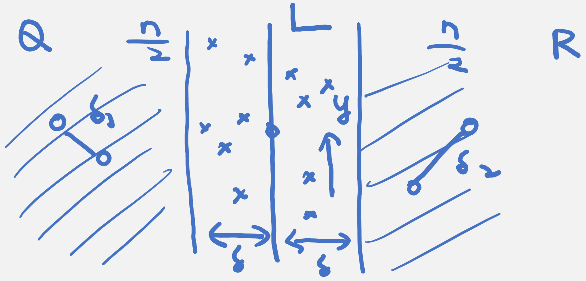
Obs 1 + Obs 2 ⇒ at most 11 other "candidate" points with distance < δ

⇒ each point computes distance with O(1) points

⇒ search space O(n)



Algorithm



$\delta = \min(\delta_1, \delta_2)$

1. Find the dividing line L by computing the median using the x-value. Time: $O(n)$.

2. Recursively solve the closeset pair problem in Q and in R. Get δ. Time: $2T(\frac{n}{2})$.

3. Using a linear scan, remove all the points not within the narrow region defined by δ. Time: $O(n)$.

4. Sort the points in non-decreasing order by their y-value. Time: $O(n \log n)$.

5. For each point, we compute its distance to the next eleven points in this y-ordering. Time: $O(n)$.

   (Note that two points within two layers must be within 11 points in the y-order, as ≤ 10 boxes in between.)

for $1 \le i \le n$
 for $i+1 \le j \le i+11$
  distance (p_i, p_j)

6. Return the minimum distance found.

Time Complexity

sorting

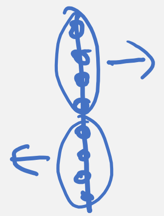
$$T(n) = 2T\left(\frac{n}{2}\right) + O(n\log n) \Rightarrow T(n) = O(n\log^2 n)$$

Sort by y-values once in the beginning. use this ordering for step 5

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O(n\log n)$$

don't need to compute median inside the recursion

sort by x-values once in the beginning. use this ordering for step 1



Question: Where did we use the assumption that all the x-values are distinct?

Question: Where do we need to change so that the algorithm would work without this assumption?

Remark: There is a randomized algorithm with $O(n)$ expected time. See [KT 13.7].

Today’s Plan

1. Closest Pair

2. Arithmetic Problems

Integer Multiplication

Input: two n-bit numbers $a = a_1 a_2 \dots a_n$ and $b = b_1 b_2 \dots b_n$.

$a_i \in \{0,1\}$ $\quad b_j \in \{0,1\}$

Output: the product ab

$$
\begin{array}{r}
1101 \\
\times \quad 1001 \\
\hline
1101 \\
0000 \\
0000 \\
1101 \\
\hline
1110101
\end{array}
$$

$\Theta(n^2)$ bit operations

Divide and Conquer

Suppose we know how to multiply n-bit numbers. Now we want to multiply $2n$-bit numbers.

$$x = \underbrace{x_1}_{n\text{ bits}} \underbrace{x_2}_{n\text{ bits}} \qquad 2n\text{-bit number} \qquad y = \underbrace{y_1}_{n\text{ bits}} \underbrace{y_2}_{n\text{ bits}}$$

$$x = x_1 \cdot 2^n + x_2 \qquad\qquad y = y_1 \cdot 2^n + y_2$$

$$T(2n) \nearrow \; xy = (x_1 \cdot 2^n + x_2) \cdot (y_1 \cdot 2^n + y_2)$$

$$= \underbrace{x_1 y_1}_{T(n)} 2^{2n} + (\underbrace{x_1 y_2}_{T(n)} + \underbrace{x_2 y_1}_{T(n)}) 2^n + \underbrace{x_2 y_2}_{T(n)}$$

$$T(2n) = 4T(n) + O(n) \leftarrow \text{shifting \& additions}$$

$$\Rightarrow \quad T(n) = O(n^2).$$

Karatsuba's Algorithm

A clever way to combine only 3 subproblems.

$$xy = x_1 y_1 2^{2n} + (x_1 y_2 + x_2 y_1) 2^n + x_2 y_2$$

3 subproblems : $x_1 y_1$, $x_2 y_2$, $(x_1 + x_2) \cdot (y_1 + y_2)$ Time : $3T(\frac{n}{2})$

middle coefficient : $(x_1 + x_2) \cdot (y_1 + y_2) - x_1 y_1 - x_2 y_2$

$$= \cancel{x_1 y_1} + \underbrace{x_2 y_1 + x_1 y_2}_{\text{middle}} + \cancel{x_2 y_2} - \cancel{x_1 y_1} - \cancel{x_2 y_2}$$

Time : $O(n)$

time complexity : $T(2n) = 3T(n) + O(n)$

$$\Rightarrow T(n) = O(n^{\log_2 3}) = O(n^{1.59})$$

Polynomial Multiplication

Input: two degree n polynomials $A(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ and

$$B(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0.$$

Output: their product $A \cdot B(x)$

Karatsuba $O(n^{1.58})$ word operations

Assumption: Each $a_i \cdot b_j$ can be done in one word operation.

Matrix Multiplication

Input: two $n \times n$ matrices A and B.

Output: their product AB

$$n\overset{n}{\begin{pmatrix} A \end{pmatrix}} \quad n\overset{n}{\begin{pmatrix} B \end{pmatrix}}$$

$$(AB)_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

each entry $O(n)$ time

total : $O(n^3)$.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \qquad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$A, B \in \mathbb{R}^{2n \times 2n}$

$T(n)$

$$AB = \begin{pmatrix} \underline{A_{11} B_{11}} + \underline{A_{12} B_{21}} & \underline{A_{11} B_{12}} + \underline{A_{12} B_{22}} \\ \underline{A_{21} B_{11}} + \underline{A_{22} B_{21}} & \underline{A_{21} B_{12}} + \underline{A_{22} B_{22}} \end{pmatrix}$$

$A_{12} \ldots \in \mathbb{R}^{n \times n}$

$$T(2n) = 8T(n) + O(n^2) \Rightarrow T(n) = O(n^3)$$

Strassen's Algorithm

Strassen surprised the world by his magic formula.

$$AB = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

$P_3 + P_4 = A_{21}B_{11} + A_{22}B_{21}$

where $P_1 = A_{11}(B_{12} - B_{22})$, $P_2 = (A_{11} + A_{12})B_{22}$, $P_3 = (A_{21} + A_{22})B_{11}$, $P_4 = A_{22}(B_{21} - B_{11})$,

$P_5 = (A_{11} + A_{22})(B_{11} + B_{22})$, $P_6 = (A_{12} - A_{22})(B_{21} + B_{22})$, $P_7 = (A_{11} - A_{21})(B_{11} + B_{12})$.

time complexity: $T(2n) = 7T(n) + O(n^2)$ ← addition, subtraction

$\Rightarrow T(n) = O(n^{\log_2 7}) = O(n^{2.81})$

Progress and Applications

It is still an active research topic to design faster algorithms for matrix multiplication.

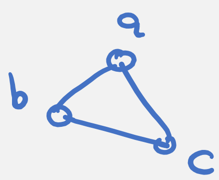
long line of work $\rightsquigarrow$ Coppersmith-Winograd $n^{2.38}$

Williams $n^{2.37}$ conjecture : $\tilde{O}(n^2)$

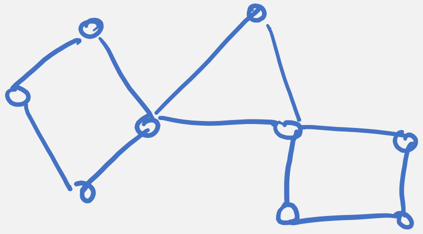
Strassen $n \gtrsim 5000$

Many combinatorial problems can be reduced to matrix multiplication to obtain fastest known algorithms.

shortest path - triangle



puzzle : use matrix multiplication
to determine if $\exists$ triangle



hint : A^2

Concluding Remarks

Fast Fourier Transform gives $O(n \log n)$-time algorithms for integer and polynomial multiplications.

interesting, praticaly useful [DPV 2.6] CS 487

Arithmetic problems are where divide-and-conquer is most powerful.