

Algorithms – Divide and Conquer II

Divide and Conquer II

Design and Analysis of Algorithms
Andrei Bulatov

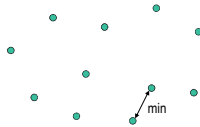
Algorithms – Divide and Conquer II 6-2

Closest Pair: The Problem

The Closest Pair Problem

Instance:
n points in the plane

Objective:
Find a pair of points that are closest together



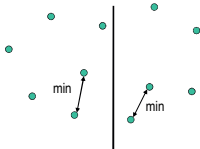
Algorithms – Divide and Conquer II 6-3

Algorithm Idea

We have an algorithm that runs in $O(n^2)$ time

Divide and conquer:

- Split the set of points into two halves
- Find closest pairs in the halves
- Check if there is a closer pair crossing the border



Algorithms – Divide and Conquer II 6-4

Assumptions and Constructions

Assumption:
No two points have the same x-coordinate or the same y-coordinate

Let P be the set of points.

We create lists P_x and P_y of points sorted by the x- and y-coordinate, respectively

This can be done using the same divide and conquer process, sorting, and then merging the two halves.

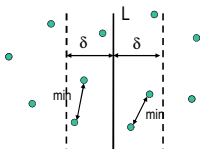
Let Q and R be the two halves,

Q_x, Q_y and R_x, R_y the halves ordered with respect to the x- and y-coordinates

Let q_0^*, q_1^* and r_0^*, r_1^* be the closest pairs from the two halves

Algorithms – Divide and Conquer II 6-5

Crossing the Border



L is given by $x = x^*$

Let δ be the minimum of $d(q_0^*, q_1^*)$ and $d(r_0^*, r_1^*)$

Lemma

If there exist $q \in Q$ and $r \in R$ such that $d(q, r) < \delta$, then each of q and r lies within a distance δ of L

Algorithms – Divide and Conquer II 6-6

Crossing the Border (cntd)

Proof

Suppose q and r exist, say, $q = (q_x, q_y)$ and $r = (r_x, r_y)$

We have $q_x \leq x^* \leq r_x$

Then

$$x^* - q_x \leq r_x - q_x \leq d(q, r) < \delta$$

and

$$r_x - x^* \leq r_x - q_x \leq d(q, r) < \delta$$

so each of q and r has an x-coordinate within δ of x^* and hence lies within distance δ of the line L

QED

Crossing the Border (cntd)

Let S denote the set of points belonging to the band of width δ around L

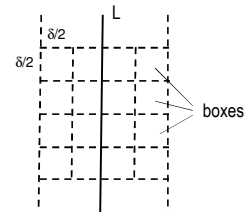
Let also S_y be the set S sorted by increasing y -coordinate

Lemma

There exist $q \in Q$ and $r \in R$ for which $d(q,r) < \delta$ if and only if there exist $s, s' \in S$ for which $d(s,s') < \delta$

Crossing the Border (cntd)**Lemma**

If $s, s' \in S$ have the property that $d(s,s') < \delta$, then s and s' are within 15 positions of each other in the sorted list S_y

**Crossing the Border (cntd)****Proof**

Denote by Z the band of width δ around L

We partition Z into boxes: squares with side $\delta/2$

No two points belong to the same box, as it contradicts the assumption that minimal distance between two points on the same side of L is δ

Since the distance between s, s' is less than δ they also cannot be more than 2 boxes apart vertically

QED

Algorithm

Closest-Pair(P)

construct P_x and P_y /*In $O(n \log n)$ time
set $(p_0, p_1) := \text{Closest-Pair-Rec}(P_x, P_y)$

Algorithm

Closest-Pair-Rec(P_x, P_y)

if $|P| \leq 3$ then use brute force

construct Q_x, Q_y, R_x, R_y /* $O(n)$ time

$(q_0, q_1) := \text{Closest-Pair-Rec}(Q_x, Q_y)$

$(r_0, r_1) := \text{Closest-Pair-Rec}(R_x, R_y)$

set $\delta := \min\{d(q_0, q_1), d(r_0, r_1)\}$

set $x^* := \max$ x-coord. of points in Q , $L := \{x = x^*\}$

set $S := \{\text{points in } P \text{ within distance } \delta \text{ from } L\}$

construct S_y

for each $s \in S$ compute dist. to next 15 points in S_y

if s, s' is the pair achieving the minimum and

$d(s, s') < \delta$ return (s, s')

else if $d(q_0, q_1) < d(r_0, r_1)$ then return (q_0, q_1)

else return (r_0, r_1)

Closest Pair: Analysis**Theorem**

The Closest-Pair algorithm outputs a closest pair of points in P

Theorem

The Closest-Pair algorithm runs in $O(n \log n)$ time for a list of n elements

Integer Multiplication

How much time do we really need to multiply two numbers?

Standard algorithm

$$\begin{array}{r} 1101 \\ 1011 \\ \hline 1101 \\ 1101 \\ \hline 1101 \\ 10001111 \end{array}$$

takes $O(n^2)$ time

Divide-and-Conquer Algorithm

Let x and y be given numbers, n digits each.

Represent them as: $x = x_1 \cdot 2^{n/2} + x_0$, $y = y_1 \cdot 2^{n/2} + y_0$

Then

$$\begin{aligned} xy &= (x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0) \\ &= x_1 y_1 \cdot 2^n + (x_1 y_0 + x_0 y_1) \cdot 2^{n/2} + x_0 y_0 \end{aligned}$$

We need to compute 4 smaller products

Does it help? Let $T(n)$ be the running time

$$T(n) \leq 4T(n/2) + Cn$$

By the Master Theorem

$$T(n) \leq O(n^{\log 4}) = O(n^2)$$

Divide-and-Conquer Algorithm (cntd)

Reduce the number of recursive calls

$$xy = x_1 y_1 \cdot 2^n + (x_1 y_0 + x_0 y_1) \cdot 2^{n/2} + x_0 y_0$$

Compute $(x_1 + x_0)(y_1 + y_0)$ and then

$$x_1 y_0 + x_0 y_1 = (x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0$$

Does it help?

Now we need 3 recursive calls: $(x_1 + x_0)(y_1 + y_0)$, $x_1 y_1$, $x_0 y_0$

Thus $T(n) \leq 3T(n/2) + Cn$

By the Master Theorem $T(n) \leq O(n^{\log 3}) = O(n^{1.59})$