CSS.201.1 Algorithms

Instructor: Umang Bhaskar

TIFR 2024, Aug-Dec

Scribe: Soham Chatterjee

sohamchatterjee999@gmail.com
Website: sohamch08.github.io

Contents

CHAPTER 1

Finding Closest Pair of Points

Problem: Given a set of points find the closest pair of points in $\mathbb{R}^2$.
Input: Set $S = \{(x_i, y_i) \mid x_i, y_i \in \mathbb{R},\ \forall\ i \in [n]\}$. We denote $P_i = (x_i, y_i)$.
Output: P_i, P_j that are at minimum l_2 distance i.e. minimize $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

1.1 Naive Algorithm

Now the naive algorithm for this will be checking all pairs of points and take their distance and output the minimum one. There are total $\binom{n}{2}$ possible choices of pairs of points. And calculating the distance of each pair takes $O(1)$ time. So it will take $O(n^2)$ times to find the closest pair of points.
Idea: $\forall\ P_i, P_j \in S$ find distance $d(P_i, P_j)$ and return the minimum. Time taken is $O(n^2)$.

1.2 Divide and Conquer Algorithm

Definition 1.2.1: Divide and Conquer

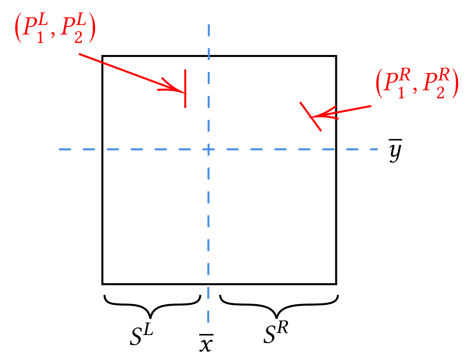
- Divide: Divide the problem into two parts (roughly equal)
- Conquer: Solve each part individually recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
- Combine: Combine the solutions to the subproblems into the solution.

1.2.1 Divide

So to divide the problem into two roughly equal parts we need to divide the points into two equal sets. That we can do by sorting the points by their $x-$coordinate. Suppose S^x denote we get the new sorted array or points. And similarly we obtain S^y which denotes the array of points after sorting S by their $y-$coordinate.

Algorithm 1: Step 1 (Divide)

Function *Divide*:
  Sort S by $x-$coordinate and $y-$coordinate
  $S^x \longleftarrow S$ sorted by $x-$coordinate
  $S^y \longleftarrow S$ sorted by $y-$coordinate
  $\bar{x} \longleftarrow \lfloor \frac{n}{2} \rfloor$ highest $x-$coordinate
  $\bar{y} \longleftarrow \lfloor \frac{n}{2} \rfloor$ highest $y-$coordinate
  $S^L \longleftarrow \{P_i \mid x_i < \bar{x},\ \forall\ i \in [n]\}$
  $S^R \longleftarrow \{P_i \mid x_i \geq \bar{x},\ \forall\ i \in [n]\}$



1.2.2 Conquer

Now we will recursively get pair of closest points in S_L and S_R. Suppose the (P_1^L, P_2^L) are the closest pair of points in S^L and (P_1^R, P_2^R) are the closest pair of points in S^R.

Algorithm 2: Step 1 (Solve Subproblems)

Function *Conquer*:
- Solve for S_L, S^R.
- (P_1^L, P_2^L) are closest pair of points in S_L.
- (P_1^R, P_2^R) are closest pair of points in S_R.
- $\delta^L = d(P_1^L, P_2^L)$, $\delta^R = d(P_1^R, P_2^R)$
- $\delta_{min} \longleftarrow \min\{\delta^L, \delta^R\}$

1.2.3 Combine

Now we want to combine these two solutions.

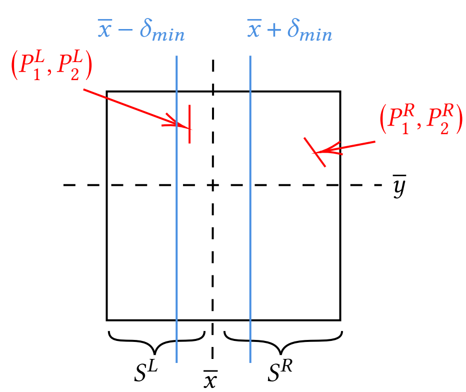
Question 1.1: We are not done

Is there a pair of points $P_i, P_j \in S$ such that $d(P_i, P_j) < \delta_{min}$

If Yes:

- One of them must be in S_L and the other is in S_R.
- $x-$coordinate$\in [\overline{x} - \delta_{min}, \overline{x} + \delta_{min}]$.
- $|y_i - y_j| \le \delta_{min}$

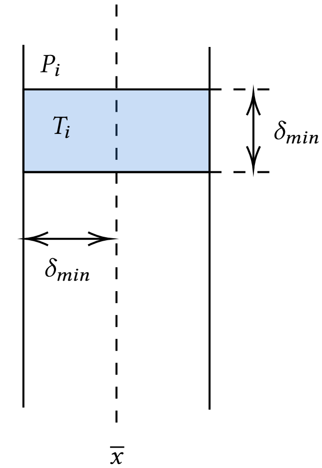
So we take the strip of radius δ_{min} around $\overline{x}$. Define $T = \{P_i \in S \mid |x_i - \overline{x}| \le \delta_{min}\}$



We now sort all the points in the T by their decreasing $y-$coordinate. Let T_y be the array of points. For each $P_i \in T_y$ define the region

$$T_i = \{P_j \in T_y \mid 0 \le y_j - y_i \le \delta_{min}, j > i\}$$

> **Lemma 1.2.1**
>
> Number of points (other than P_i) that lie inside the box is at most 8



Proof: Suppose there are more than 8 points that lie inside the box apart from P_i. The box has a left square part and a right square part. So one of the squares contains at least 5 points. WLOG suppose the left square has at least 5 points. Divide each square into 4 parts by a middle vertical and a middle horizontal line. Now since there are 5 points there is one part which contains 2 points but that is not possible as those two points are in S_L and their distance will be less than δ_{min} which is not possible. Hence contradiction. Therefore there are at most 8 points inside the box. ■

Hence by the above lemma for each $P_i \in T_y$ there are at most 8 points in T_i. So for each $P_j \in T_i$ we find the $d(P_i, P_j)$ and if it is less than δ_{min} we update the points and the distance

1.2.4 Pseudocode and Time Complexity

Algorithm 3: FIND-CLOSEST(S)

Input: Set of n points, $S = \{(x_i, y_i) \mid x_i, y_i \in \mathbb{R},\ \forall\, i \in [n]\}$. We denote $P_i = (x_i, y_i)$.
Output: Closest pair of ponts, (P_i, P_j, δ) where $\delta = d(P_i, P_j)$

begin
  if $|S| \leq 10$ **then**
    Solve by Brute Force (Consider every pair of points)
  $S^x \longleftarrow S$ sorted by $x-$coordinate
  $S^y \longleftarrow S$ sorted by $y-$coordinate
  $\bar{x} \longleftarrow \lfloor \frac{n}{2} \rfloor$ highest $x-$coordinate
  $\bar{y} \longleftarrow \lfloor \frac{n}{2} \rfloor$ highest $y-$coordinate
  $S^L \longleftarrow \{P_i \mid x_i < \bar{x},\ \forall\, i \in [n]\}$
  $S^R \longleftarrow \{P_i \mid x_i \geq \bar{x},\ \forall\, i \in [n]\}$
  $(P_1^L, P_2^L, \delta^L) \longleftarrow$ FIND-CLOSEST(S^L)
  $(P_1^R, P_2^R, \delta^R) \longleftarrow$ FIND-CLOSEST(S^R)
  $\delta_{min} \longleftarrow \min\{\delta^L, \delta^R\}$
  if $\delta_{min} < \delta^L$ **then**
    $P_1 \longleftarrow P_1^R,\ P_2 \longleftarrow P_2^R$
  else
    $P_1 \longleftarrow P_1^L,\ P_2 \longleftarrow P_2^L$

Chapter 2

Median Finding

Chapter 3

Bibliography