Closest Points Problem

Input list P of n points on Euclidean plane:

\[ P_i = (x_i, y_i), \quad P_2 = (x_2, y_2), \ldots, \quad P_n = (x_n, y_n) \]

Goal: Find the closest pair of points.

Distance b/w 2 points \( P_i, P_j \):

\[ d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \]

Applications: Air/sea traffic controllers use to detect closest vehicles to avoid collisions.

For simplicity, we will find min distance (instead of points):

Brute force in points, so \( n(n-1)/2 \) pairs.

Check all distances:

\[ \Theta(n^2) \]

How to do better?

Intuitively:

Probably shouldn't check these \( \circ \) points.

\( \circ \)
How to account for this?

- Partition the points: left, right

\[ \begin{array}{c}
\text{P} \\
\text{PL} \quad \text{PR}
\end{array} \]

Now what? Same problem within PL, PR:

\[ \begin{array}{c}
\text{0} <- \text{no need to compare} \\
\text{PL}
\end{array} \]

So, continuously (recursively) split each subspace in half.

When to stop recursing? When subspace contains exactly 3 points.

Now what?

- Find distance b/w pair in PL
- " " " " " " PR
- " " " " all pairs in PL, PR (crossing)
- Find min distance of these 3"
Preprocess (P)

Sort P by x-coord.

Closest (P) // P, E1, nJ : list of n points
     // Finds distance bw closest pair in P.

if (|P| = ³√n) return dist(p1,p2) //using distance formula
    min distance of all pairs.
else

    PL, PR = Partition (P) // partition P into left, right

    dL = Closest (PL) //closest distance on left

    dR = Closest (PR) // " " " right

    d = min (dL, dR, de)

    de = closestCrossing (PL, PR, d) //closest dist
        //among crossing

    <What to eventually return?> //pairs.

return min (dL, dR, de)

Questions:
(1) How to partition?
(2) What if subspace contains only 4 points?
(3) How to find closest crossing distance in less than O(n²) time
(1) Partition: "How to partition into $P_L, P_R$?"

**Partition ($P$)**

- sort $P$ by x-coord
- $P_L = P[1 \ldots \lfloor n/2 \rfloor]$
- $P_R = P[\lceil n/2 \rceil + 1 \ldots n]$

return $P_L, P_R$.

**Problem?** Partition() called for every recursive call
So sort required each time.

**Solution**: Sort $P$ once at the beginning!

```plaintext
< Preprocess ($P$) >  \leftarrow add to Closest() code
  Sort $P$ by x-coord  \rightarrow remove from Partition()
```

(2) "What if subset contains only 1 point?"
Suppose subset has 5 points.

\[ r_1, r_2, r_3 \rightarrow \text{compute using dist formula} \]

2nd partition 1st partition

Can't compute distance for single point!

So set Base Case at \( n = 3 \)

Another solution:

Have 2 Base Cases:

- if \( n = 1 \) return \( \infty \)
- if \( n = 2 \) return \( \text{dist}(p_1, p_2) \)

⇒ Change Code!

Question (3): Closest Crossing:

(Step back to see how algorithm works)

\[ p_L, p_R, p_1, p_2 \]

How to find closest distance b/w crossing pairs?
Simplest approach? For every point in \( P_L \), compute its distance to every point in \( P_R \).

Closest crossing \( (P_L, P_R, d) \): // computes min distance b/w pairs of points crossing \( P_L \) \& \( P_R \)

<Will change the code a lot!>

for each \( p_i \) in \( P_L \):
- compute distance b/w \( p_i \) and every point in \( P_R \)
  <Leave a lot of space>

return min dist

\[ \text{Max # of computations? } \frac{n}{2} \times \frac{n}{2} = O(n^2) \]

"We want to do better than \( O(n^2) \), maybe \( O(n) \)."

"Do we need to compute distance b/w \( p_i \) and every point in \( P_R \)?"

\( \circ \) No! Again, probably not necessary to compute distance b/w these 2.

\( p_i \)

"Which points do we care about?"
"Hint: Look back at Closest()"

We know closest distance on left $d_L$
We know " " " right $d_R$

"So we know" $\min(d_L, d_R)$ is closest distance of non-crossing points!

Add to code: $d = \min(d_L, d_R)$ in Closest() send $d$ as parameter to ClosestCrossing()

"So compare $P_1$ only to points that are within $d$ of $P_1$"

"But finding such points is tricky (may still be $O(n^2)$, so any other way to discard some points?"

Hint:

1. $d_R = 20$
2. $d_L = 10$

$\Rightarrow$ can discard (distance to any point in $P_L$ will be $>10$)!

<Update ClosestCrossing>
for each $p_i$ in $P_e$:

- Compute distance b/w $p_i$ and every point in $P_R$ that is within horizontal distance $d$ from partition.

  Get min distance.

"How to find points within horizontal distance $d$ from partition?"

Points already sorted by $x$-coord so just check if $x$-coord is within $d$ of $x$-coord of partition.
\[(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{2n}, y_{2n})\]

\[x\text{-coord of partition} = \frac{x_{n/2} + x_{n/2+1}}{2} = x_{\text{part}}\]

In \text{ClosestCrossing()}:

for each \(P_i\) in \(P_L\):

- compute distance b/w \(P_i\) and every point \(P_j\) in \(P_L\) such that \(|x_2 - x_{\text{part}}| \leq d\).

return min dist

\[\text{Now, maximum number of comparisons?} \]
\[\frac{n}{2} \times \frac{n}{2} = O(n^2) \text{ Still}\]

\[\text{ex: All points clustered around partition}\]

\[\text{Should compare } p \text{ and } b, \text{ but not } p \text{ and } a!\]

So we need to also consider y-coords.

\[\text{If vertical dist } > d, \text{ don't compare!}\]
<Update Closest Crossing>

for each $p_1$
- compute distance b/w $p_1$ and every point $p_2$
in $P_2$ such that:
  - $|x_2 - x_{\text{part}}| < d$ and
  - $|y_2 - y_1| < d$

Now, how many computations?

Turns out to be $O(n)$.

For every point $p_1$, I will compute distance to
only 15 other points!
Example partition at 18, d=10

Need only consider points within p's box
How many such points?

Split p's box into 16 regions.
each region: \[ \frac{d}{2} \]

Max per region? \[ \frac{d}{2} \]

* What do we know about the points left/right of partition? *

* All pairs left/right are \( \geq d \) *

- How does this help us? 

Let's try to put 2 in one region.

Farthest we can place 2 is at corners:

\[
\begin{cases}
- \frac{d}{2} \\
\frac{d}{2}
\end{cases}
\]

Are they \( \geq d \) apart?

\[
dist = \sqrt{\left(\frac{d}{2}\right)^2 + \left(\frac{d}{2}\right)^2} = \sqrt{\frac{2d^2}{4}} = \sqrt{\frac{d^2}{2}} = \frac{d}{\sqrt{2}} > 1
\]

- At most 1 point in each region

- 15 points to compare to \( \rho \)

Total \# comparisons in \text{ClosestCrossing}():

For every point in \( P \), that is within horizontal distance \( d \) from partition \( S / 2 \)

- compare to 15 other points

\[
\Rightarrow \frac{n}{2} \times 15 = \frac{n}{8}
\]

Much better than \( \frac{n^2}{4} \)!