Closest Pairs Divide-and-Conquer (CPDQ)

David Liu

The Algorithm

```
def CPDQ(A):
   
   # Pre: A is a list of 2D points [(x1, y1), ..., (xn, yn)]
   # Post: Returns the length of the smallest distance between
   # two points in A
   
   if len(A) <= 3:
      return bruteForce(A)
   else:
      # DIVIDE step
      sortedA = sortByX(A) # Sorts points by x-coordinate
      m = len(A) // 2
      L = sortedA[0..m-1] # Length is n/2
      G = sortedA[m..len(A)-1] # Length is n/2

      # RECURSIVE step
      minL = CPDQ(L)
      minG = CPDQ(G)

      # COMBINE step
      middle = average of x-coords of L[m-1] and G[0]
      d = min(minL, minG)
      L_d = points in L within d of the middle
      G_d = points in G within d of the middle

      L_d = sortByY(L_d) # Sorts points by y-coordinate, largest first
      G_d = sortByY(G_d) # Sorts points by y-coordinate, largest first

      # I'm expanding this code a little bit to show why it's linear time
      # Notice the similarity to the merge function in mergesort.
      i = 0
      j = 0
      while i < m and j < m:
         if L_d[i].y >= G_d[j].y: # Comparing y-coordinates
            check distances between (L_d[i], G_d[j]) and (L_d[i], G_d[j+1])
            update d if necessary
            i += 1
         else:
            check distances between (L_d[i], G_d[j]) and (L_d[i+1], G_d[j])
            update d if necessary
            j += 1
      return d
```
The Proof

This is a proof that for every point in L₅d and G₅d, we only needed to check the next two points. The content is more mathematically advanced that is required for this course; treat this as enrichment!

To make the math easier, we assume the points are ordered from lowest to highest; also, we start with a point in L₅d. The same argument basically applies to our algorithm (by flipping it upside-down), and for starting points in G₅d as well. Let P be a point in L₅d. Without loss of generality, we assume the following:

- The value of d (in the code) is 1
- The value of middle is 0
- The coordinates of P are (−x, 0) (so 0 ≤ x ≤ 1)

We first prove the following lemma.

Lemma 1. Let P₁ = (a₁, b₁), P₂ = (a₂, b₂) be two points in G₅d above P, with 0 ≤ b₁ ≤ b₂. Also, suppose a₁ ≤ 1/2. If the distance between P and P₂ is ≤ 1, then P₁ is closer to P than P₂.

Proof. First, if a₂ ≥ a₁, then P₂ is certainly farther from P than P₁, and the claim is true. So for the rest of this proof, we’ll assume a₂ < a₁. Because P₁ and P₂ are both in G, the distance between them is at least d = 1:

\[(b₂ - b₁)^2 + (a₂ - a₁)^2 ≥ 1 \]
\[(b₂ - b₁)^2 ≥ 1 - (a₂ - a₁)^2 \]
\[(b₂ - b₁) ≥ 1 - a₁^2 \quad \text{(since } 0 ≤ a₂ < a₁)\]

b₂ ≥ b₁ + √1 - a₁²

Now let us calculate the difference in their respective distances from P:

\[(b₂^2 + (a₂ + x)^2) - (b₁^2 + (a₁ + x)^2) = b₂^2 - b₁^2 + (a₂ + x)^2 - (a₁ + x)^2 \]
\[≥ 2b₁\sqrt{1 - a₁²} + 1 - a₂^2 + (a₂ + x)^2 - (a₁ + x)^2 \]
\[≥ 1 - a₁² - (a₁ + a₂ + 2x)(a₁ - a₂) \quad \text{(since } b₁ ≥ 0, \text{ and difference of squares)}\]
\[≥ 1 - a₁² - a₁(a₁ + 2x) \quad \text{(minimum achieved when } a₂ = 0)\]

Assuming that the distance between P and P₂ is ≤ 1, we must have x ≤ a₁ (because b₂ ≥ √1 - a₁²). Therefore

1 - a₂² - a₁(a₁ + 2x) ≥ 1 - a₁² - a₁(3a₁) ≥ 1 - 4a₁² ≥ 0,

where the last inequality comes from the fact that a₁ ≤ 1/2. Thus P₂ is farther from P than P₁. ■

Our second claim covers the other case, when we start with a point further than 1/2 from the middle.

Lemma 2. Let P₁ = (a₁, b₁), P₂ = (a₂, b₂), P₃ = (a₃, b₃) be three points in G₅d above P: so 0 ≤ b₁ ≤ b₂ ≤ b₃. Also, suppose a₁ ≥ 1/2. If the distance between P and P₃ is ≤ 1, then P₁ or P₂ (or both) is closer to P than P₃.

Proof. First, note that if a₂ ≤ 1/2, then by Lemma 1 applied to P₂ and P₃, P₂ is closer to P than P₃. So we’ll assume that 1/2 < a₂ ≤ 1.

We claim that b₃ ≥ b₁ + √2a₁ - a₁², which is the y-value of the intersection of the circle of radius 1 centred on P₁ with the line y = 1. From a similar calculation as in the proof of Lemma 1, b₂ ≥ b₁ + √1 - (a₂ - a₁²), because P₁ and P₂ are at least 1 unit apart. Now we consider the possible location of P₃, constrained by being at least 1 unit from P₂:

\[(a₃ - a₂)^2 + (b₃ - b₂)^2 ≥ 1 \]
\[a₂^2 + (b₃ - b₂)^2 ≥ 1 \quad \text{(since } a₃ ≥ 0)\]
\[b₃ ≥ b₂ + √1 - a₂² \]
\[b₃ ≥ b₁ + √1 - (a₂ - a₁²)^2 + √1 - a₂² \]
Thus it suffices to show that \( \sqrt{1 - (a_2 - a_1)^2} + \sqrt{1 - a_2^2} \geq \sqrt{2a_1 - a_1^2} \). We have:

\[
\begin{align*}
1 + a_2 &\geq a_1 \\
1 - a_2^2 &\geq a_1 - a_1 a_2 \\
1 - a_2^2 + a_1 a_2 &\geq a_1 \\
2 - 2a_2^2 + 2a_1 a_2 &\geq 2a_1 \\
1 - (a_2 - a_1)^2 &+ 1 - a_2^2 \geq 2a_1 - a_1^2 \\
\sqrt{1 - (a_2 - a_1)^2} + 1 - a_2^2 &\geq \sqrt{2a_1 - a_1^2} \\
\sqrt{1 - (a_2 - a_1)^2} + \sqrt{1 - a_2^2} &\geq \sqrt{2a_1 - a_1^2}
\end{align*}
\]  

((x + y) \leq \sqrt{x} + \sqrt{y} for all \( x, y \geq 0 \))

Thus we've shown \( b_3 \geq b_1 + \sqrt{2a_1 - a_1^2} \). Now we compare the distances between \((P, P_1)\) and \((P, P_3)\):

\[
\begin{align*}
(b_3^2 + (a_3 + x)^2) - (b_1^2 + (a_1 + x)^2) &= b_3^2 - b_1^2 + (a_3 + x)^2 - (a_1 + x)^2 \\
&\geq 2a_1 - a_1^2 - (a_1 + a_3 + 2x)(a_1 - a_3) &\text{(similar calculation)} \\
&\geq 2a_1 - a_1^2 - a_1(a_1 + 2x) &\text{\((a_3 \geq 0)\)} \\
&\geq 2a_1 - a_1^2 - a_1(2(1 - a_1)) &\text{\(x \leq 1 - a_1\) because \(b_3 \geq \sqrt{1 - (1 - a_1)^2}\)} \\
&= 2a_1 - a - 1^2 - a_1(2 - a_1) \\
&= 0
\end{align*}
\]

So in this case, \(P_1\) is closer to \(P\) than \(P_3\).

Combining the previous two lemmas immediately yields the following theorem.

**Theorem 1.** Let \(P\) be a point in \(L_d\). If some point in \(G_d\) is closer than 1 unit to \(P\), then the closest point in \(G_d\) to \(P\) is one of the first two points in \(G_d\) above \(P\).

The Next Steps

1. The code provided on the first page runs in time \(O(n \log n)^2\). It does some redundant work; can you simplify this algorithm to run in time \(O(n \log n)\)?

2. Also, the proof that you only need to check the next two points in \(G_d\) is messier than I thought it would be; can you simplify the proof?