CSE 241 Algorithms and Data Structures 8/31/2015

Fast Closest-Pair Algorithm

This handout gives pseudocode for the $\Theta(n \log n)$ closest-pair algorithm in the form I presented in class. In this handout and all future ones, I'm pretty sure the pseudocode is correct, but I make no iron-clad guarantees – if something looks wrong to you, think before you implement!

Keep in mind that the arrays ptsByX and ptsByY in the algorithm (a) are assumed to be sorted by x and y coordinate respectively and (b) are actually *references* to a common set of points, *not* two distinct point sets. n is the number of points.

```
ClosestPair(ptsByX, ptsByY, n)
  if n = 1
     return ∞
  if n = 2
     return distance(ptsByX[0], ptsByX[1])

  mid ← ⌈n/2⌉ − 1                                  ▷ divide into two subproblems
  copy ptsByX[0 . . . mid] into new array XL in x order.
  copy ptsByX[mid+1 . . . n − 1] into new array XR in x order.

  copy ptsByY into arrays YL and YR in y order, s.t.
       XL and YL refer to same points, as do XR and YR

  distL ← ClosestPair(XL, YL, ⌈n/2⌉)                                ▷ conquer
  distR ← ClosestPair(XR, YR, ⌊n/2⌋)

  midPoint ← ptsByX[mid]                                            ▷ combine
  lrDist ← min(distL, distR)
  Construct array yStrip, in increasing y order, of all
       points p in ptsByY s.t. |p.x − mid.x| < lrDist

  minDist ← lrDist
  for j in 0  . . .  yStrip.length − 2 do
     k ← j + 1
     while k ≤ yStrip.length − 1 and
              yStrip[k].y − yStrip[j].y < lrDist do
        d ← distance(yStrip[j], yStrip[k])
        minDist ← min(minDist, d)
        k++
  return minDist
```