An Easier Version

Given: A list of points in 1-dimension
Return: The distance between the two points that are closest to each other.

Your input will be a list of `doubles` (in no particular order).

What's your algorithm?

Pseudocode

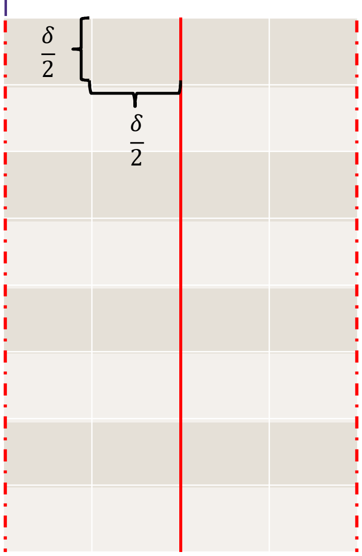
double 2DClosestPoints(P[1..n])
    if($n \leq 100$) //pick a cutoff you like; doesn't matter for big-O
        check all possible pairs, return the smallest distance.
    Sort P[] by x-coordinate
    δ ←min{2DClosestPoints(P[1..n/2]), 2DClosestPoints(P[n/2+1,n])}
    //TODO: conquer

Some Questionable Pseudocode

double 2DClosestPoints(P[1..n])
    if($n \leq 100$) //pick a cutoff you like; doesn't matter for big-O
        check all possible pairs, return the smallest distance.
    Sort P[] by x-coordinate
    δ ←min{2DClosestPoints(P[1..n/2]), 2DClosestPoints(P[n/2+1,n])}
    for(i from 1 to $n/2$)
        for(j from $n/2$ +1 to n)
            $\delta \leftarrow$ min{δ, `dist(P[i],P[j])`}
    return δ

Prove the Lemma

If `dist(P[i],P[j])` $\leq \delta$ and `P[i]`, `P[j]` are both in the middle strip, then $|i-j| \leq 11$



Place a grid of $\delta/2$x$\delta/2$ squares on the strip.

Strip is 4 squares wide.