Algorithm Closest_pair(PX, n)

//Input: An array PX[0..n-1] of points, and parameter n, size of the array. Each cell PX[i] has 3 components:
PX[i] = { PX[i].ID, PX[i].x, PX[i].y }: the ID, x and y coordinate of point PX[i]
The points in PX are sorted (ascending order) according to the x coordinate

//Output: min_dist: minimum distance between two points in PX
PY[0..n-1]: Points in PX now sorted (ascending order) according to the y coordinate

//check the base case
If n = 1 return (INF, PX)

If n = 2

   min_dist ← dist(PX[0], PX[1])
   If PX[0].y ≤ PX[1].y
      PY[0] ← PX[0]
      PY[1] ← PX[1]
   Else PY[0] ← PX[1]
      PY[1] ← PX[0]

   Return (min_dist, PY)

//Divide into two subproblems

mid ← n/2
PXL ← PX[0..mid-1]
PXR ← PX[mid .. n-1]

// Conquer the subproblems

(dL, PYL) ← Closest_pair(PXL, mid)  // the left half of the points
   // PYL is the array of points in PXL sorted by y coordinate
(dR, PYR) ← Closest_pair(PXR, n-mid)  // the right half of the points
   // PYR is the array of points in PXR sorted by y coordinate

//Combine the solutions for the subproblems

d ← min (dL, dR)  // d is the current minimum distance
PY ← merge (PYL, PYR)  // PY is the array of points in PX sorted by the y coordinate
// The merge method is essentially the same as in merge sort

mid_x ← PX[mid].x       // the x value of the split point

length ← 0
i ← 0

While (i ≤ n-1) Do

    If | PY[i].x - mid_x | ≤ d
        Strip[length] ← PY[i]       // point PY[i] is within the strip of width 2d centered around
        length ← length + 1
        // the line x = mid_x

    i ← i+1

// Now the array Strip contains all points in PY which are within
// the strip of width 2d centered around the line x = mid_x

min_dist ← d
// The number of elements in Strip is length
// Next we will check the points in Strip for possible smaller
// distance than min_dist
For i ← 0 to length-2 Do

    k ← i+1       // Only check points in Strip with index larger than i

    While (k ≤ length-1 AND Strip[k].y - Strip[i].y ≤ d) Do
        new_d ← dist(Strip[i], Strip[k])
        min_dist ← min(d, new_d)       // update the current min_dist

Return (min_dist, PY)