Algorithm to find Closest pair of Points:

Let P be the set of coordinates of n points on the plane. For any pair of points $p = (x, y)$ and $q = (x', y')$, let $d(p, q) = \sqrt{(x - x')^2 + (y - y')^2}$; i.e., $d(p, q)$ is the Euclidean distance between p and q.

```
    ClosestPair(P)
    ▷ P is a set of at least two points on the plane, given by their x- and y-coordinates
    P_x := the list of points in P sorted by x-coordinate
    P_y := the list of points in P sorted by y-coordinate
    return RCP(P_x, P_y)

    RCP(P_x, P_y)
    ▷ P_x and P_y are lists of the same set of (at least two) points sorted by x- and y-coordinate, respectively
1   if |P_x| ≤ 3 then
2       calculate all pairwise distances and return the closest pair
3   else
4       L_x := first half of P_x; R_x := second half of P_x
5       m := (max x-coordinate in L_x + min x-coordinate in R_x)/2
6       L_y := sublist of P_y of points in L_x; R_y := sublist of P_y of points in R_x
7       (p_L, q_L) := RCP(L_x, L_y); (p_R, q_R) := RCP(R_x, R_y)
8       δ := min(d(p_L, q_L), d(p_R, q_R))
9       B := sublist of P_y of points whose x-coordinates are within δ of m
10      if |B| ≤ 1 then
11          if d(p_L, q_L) ≤ d(p_R, q_R) then return (p_L, q_L)
12          else return (p_R, q_R)
13      else
14          (p*, q*) := any two distinct points on B
15          for each p ∈ B (in order of appearence on B) do
16              for each of the (up to) next seven points q after p on B do
17                  if d(p, q) < d(p*, q*) then (p*, q*) := (p, q)
18          if d(p*, q*) < δ then return (p*, q*)
19          else if d(p_L, q_L) ≤ d(p_R, q_R) then return (p_L, q_L)
20          else return (p_R, q_R)
```

For the running time analysis, let $T(n)$ be the running time of $\mathrm{RCP}(P_x, P_y)$, where n is the number of points on the list P_x (which is also the number of points on the list P_y). Assume that n is a power of 2. We claim that $T(n)$ satisfies the recurrence $T(n) = 2T(n/2) + cn$, for some constant c. This is because the algorithm calls itself twice on instances of half the size (see line 7), and requires $\Theta(n)$ time to divide up the input and to combine the results of the two smaller instances into the result of the original instance. To see the latter point (i.e., that the algorithm requires only $\Theta(n)$ time for the divide and combine steps), note that lines 4, 6, and 9 can be easily implemented in $\Theta(n)$ time each. The loop in lines 15–17 takes $\Theta(n)$ time since there are $\Theta(n)$ points in B and each iteration takes $\Theta(1)$ time (since the inner for loop is executed only seven times for each point $p \in B$). All other lines only require $\Theta(1)$ time.

From the Master Theorem, we conclude that $T(n) = \Theta(n \log n)$.

If the given set P contains n points, the main procedure $\mathrm{ClosestPair}(P)$ requires $\Theta(n \log n)$ time to sort the points in P by x-coordinate and by y-coordinate in order to produce the lists P_x and P_y, and an additional $\Theta(n \log n)$ time for the call $\mathrm{RCP}(P_x, P_y)$. So the entire algorithm runs in $\Theta(n \log n)$ time.