Part 1 Implementation:

Naive Algorithm: Check every two points by a two-layer loop.

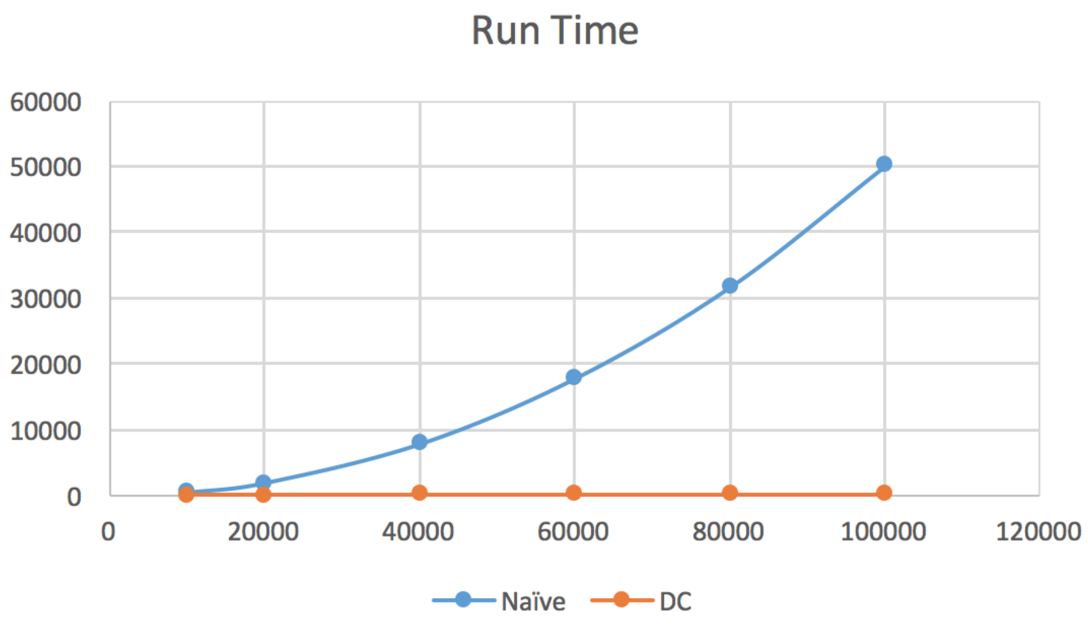
DC Algorithm: I creat a new class Result to bundle information about minDist and minPairPoints. And within it I have a public static variable which can be accessed by DC class, to set and return minDis and pairpoints. In this case, I still use the void findClosestPair() as the recurrsive function, and set DistL and DistR using results stored in type Result.

I know the other way is to create a helper method as a recursive method, and do divide and conquer within it and call from findClosestPair(). I just want to know if it is ok to use void method as the recursive method.

Part 2 Comparison:

(a)

Time (milliseconds)	10000	20000	40000	60000	80000	100000
Naive	520	1998	7921	17876	31769	50166
DC	102	132	218	218	278	293



(b) For RANDOM points passed in:

Time (milliseconds)	MIN	MAX	AVE	Variance
Naive	12326	13516	12667.15	-
DC	23	222	30.48	1269.56

For SAME points passed in:

Time (milliseconds)	MIN	MAX	AVE	Variance
Naive	12296	13747	12738.74	-
DC	25	213	32	653.16

The random points have a larger variation.

Part 3 Crossover:

When the iterration number k is becoming bigger, changing from 1000 to 1000000, the crossover number will change. It means that when we compute the average using more loops, the presision is improved. For example, when loop $k = 1000, crossover = 250$. When $k = 10000, crossover < 250$. When $k = 100000, crossover = 30$. So when the k is big enough, DC algo is definitely faster than Naive. Java will optimize the DC algo more than Naive algo.