Pseudo-code

brute force:

```
closestpair: (P_1, P_2, ..., P_n) = P
  if n < 2,
    return ∞
  else,
    min_d = d(P_1, P_2)
    closest_P = (P_1, P_2)
    for i = 1:1:n-1
      for j = i+1:1:n
        if min_d > d(P_i, P_j)
          min_d = d(P_i, P_j)
          closest_P = (P_i, P_j)
      end for
    end for
  return min_d, closest_P
```

naive divide and conquer:

```
closestpair: (XP_1, XP_2, ..., XP_n) = XP; list of points
      sorted by x-coordinate
  if n < 3,
    return bruteforce(XP)
  else,
    XP_L = XP[1: [n/2]]
    XP_R = XP[[n/2]+1 : n]
    min_d_L, closest_P_L = closestpair(XP_L)
    min_d_R, closest_P_R = closestpair(XP_R)
    if min_d_L > min_d_R,
      min_d = min_d_R
      closest_P = closest_P_R
    else,
      min_d = min_d_L
      closest_P = closest_P_L
    temp = [XP_i in XP if |XP_i - XP[[n/2]]| < min_d
    YP = sort(temp)
    if length(YP) < 2,
      return min_d, closest_P
    else,
      for i = 1:1:(length(YP)-1)
        for j = (i+1):1:length(YP)
          d = d(YP_i, YP_j)
          if d < min_d,
            min_d = d
            closest_P = (YP_i, YP_j)
    return min_d, closest_P
```

enhanced divide and conquer:

```
closest pair: same set of points XP = (XP_1, ..., X
      YP = (YP_1, ..., YP_n) sorted by x-coordinate
      and Y_coordinate
  if n < 3,
    return bruteforce(XP)
  else,
    XP_L = XP[1: [n/2]]
    XP_R = XP[[n/2]+1 : n]
    YP_L = [YP_i in YP if x-coordinate_value
            of YP_i ≤ XP[[n/2]]]
    YP_R = [YP_i in YP if x_coordinate_value
            of YP_i > XP[[n/2]]]
    min_d_L, closest_P_L = closestpair(XP_L, YP_
    min_d_R, closest_P_R = closestpair(XP_R, YP_R
    if min_d_L > min_d_R,
      min_d = min_d_R
      closest_P = closest_P_R
    else,
```

min_d = min_d_L
closest_P = closest_P_R

MYP = [YP_i in YP if the distance between X_coordinate_values of YP_i and $XP[[\frac{n}{2}]]$ is smaller than min_d]

if length(MYP) < 2,
    return min_d, closest_P

else,
    for i=1:1:length(MYP-1)
        for j=1:1:length(MYP)
            d = d(MYP_i, MYP_j)
            if d < min_d,
                min_d = d
                closest_P = (MYP_i, MYP_j)

return min_d, closest_P.

Asymtotic analysis of time

brute force:

the outer loop is n-1 time and for outer loop i, the inner loop is n-i time, so the running time will be

$(n-1)\times c + (n-2)\times c + \cdots + 2\times c + c$ (c is a constant)

$= c \cdot \frac{(n-1)n}{2}$

$= O(n^2)$

naive divide and conquer:

Sorting of X-coordinate takes $O(n\log n)$
divide and conquer takes $2T(\frac{n}{2})$
identity stripe takes $O(n)$
Sorting of Y-coordinate takes $O(n\log n)$
find min_d in middle strip takes $O(n)$ (bec 7 maximum exploration)

therefore, $T(n) = 2T(\frac{n}{2}) + cn\log n$

$n \quad cn\log n$

$\frac{n}{2} \quad \frac{n}{2} \quad 2\times c\frac{n}{2}\log\frac{n}{2} = cn\log n - c_1 n$

$\frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad 4\times c\times\frac{n}{4}\log\frac{n}{4} = cn\log n - c_2 n$

$\vdots$

base $\quad cn\log n - c_k n$

$\log n$

$T(n) = cn\log n \cdot \log n = O(n\log^2 n)$

enhanced divide and conquer:

Sorting of X-coordinate of Y-coordinate take $O(n\log n)$
divide and conquer takes $T(\frac{n}{2})$
identifying stripe takes $O(n)$
find min_d in middle strip takes $O(n)$

$T(n) = 2T(\frac{n}{2}) + cn$

$\because \log_2 2 = d = 1$

$\therefore$ from master theorem,

$T(n) = O(n\log n)$

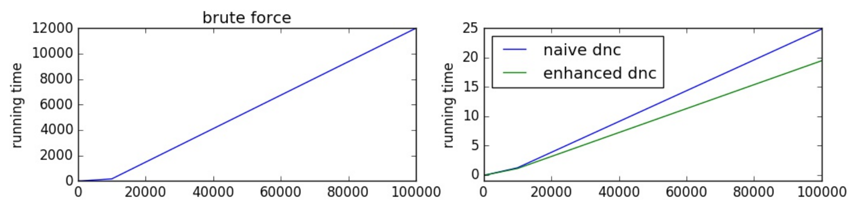
Plotting the running time

The empirical time for different for different input size is

Table 1: My caption

	100	1000	10000	100000
$brute - force$	0.01943	1.601	158.23	12000(?)
$naive - dnc$	0.003285	0.0644	1.23	24.84
$enhanced - dnc$	0.003678	0.0578	1.11	19.46

I put a ? in brute force for input size 100000 because after 3 hours I still can't get the implementation done. So I just put a 12000 there for purpose of plotting,



Interpretation and discussion

From above plot, we could see the brute force takes the most time under all input size cases. The running time for naive divide and conquer method and enhanced divide and conquer method drop sharply compared than it of brute force method. And the running time of enhanced divide and conquer method is less than it of naive divide and conquer method. The numerical values of running time does match the theoretical result. But in the plot, it is hard to say the growth curves match $O(n^2)$, $O(n \log^2 n)$ and $O(n \log n)$ respectively. I think the reason is because we don't explore enough input size cases. So the above plot is actually not a thorough representation of running time. We need explore more input sizes to see the growth curves.