Pseudo-code:

```
For i to size – 1
    Grab point 1 & point 2 and calculate distance(pt1,pt2) initialize result struct
    For j (j=i+1) to size-1
        Distance = dis (pt[i],pt[j])
        If(distance < result.distance)
            Result.distance = distance
            Clear vector which store the shortest pair of points
            Push pt[i],pt[i+1] to vector
        else if (distance == result.distance)
            push finded pairs to vector
    End for
End for
```

Asymptotic Analysis of run time:

For the out for loop it iterates through all elements of the points array who has n elements, so it will execute n times. Inner loop go through element from index i + 1 to 0, so it will go through 1 times, 2 times, … n-1 times. When out loop go through n times, total times for inner loops execution is 1 + 2 +3 + … + n-1 = (1 + n -1)*n/2 = n^2/2.

When n -> infinite, T(n) = n^2/2, which divider 2 is not import, due to the n^2 domain it. n^2/2 < n^2, so T(n) = O(n^2). The time complexity of Brute force would be O(n^2/2).

Empirical analysis and plotting:

NumInputs(100)\times	1st	2nd	3rd	4th	5th
Runtime	0.000159s	0.000149s	0.000155s	0.000170s	0.000148s
--	6th	7th	8th	9th	10th
--	0.000162s	0.000150s	0.000175s	0.000149s	0.000149s

Average: 0.000157s

NumInputs(1000)\times	1st	2nd	3rd	4th	5th
Runtime	0.014481s	0.014158s	0.014113s	0.014336s	0.014153s
--	6th	7th	8th	9th	10th
--	0.014154s	0.013996s	0.013928s	0.013956s	0.013961s

Average: 0.014248s

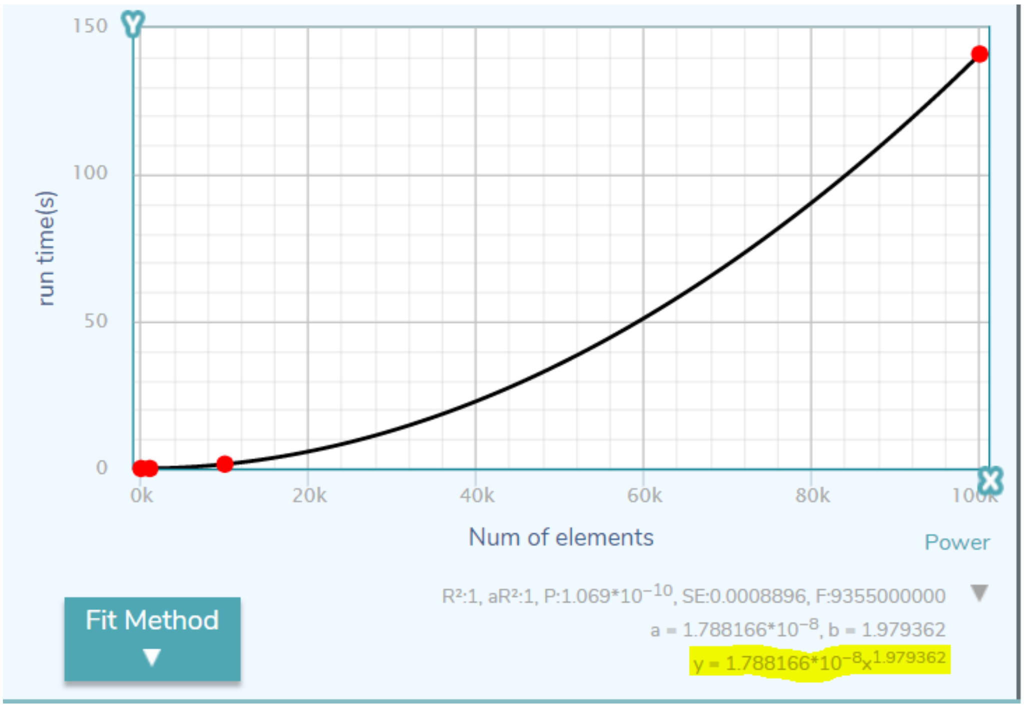
NumInputs(10000)\times	1st	2nd	3rd	4th	5th
Runtime	1.424536s	1.451141s	1.554845s	1.533729s	1.411581s
--	6th	7th	8th	9th	10th
--	1.592131s	1.407556s	1.409005s	1.589402s	1.412498s

Average: 1.4786474s

NumInputs(100000)\times	1st	2nd	3rd	4th	5th
Runtime	141s	142s	140s	140s	142s
--	6th	7th	8th	9th	10th
--	139s	140s	141s	141s	142s

Average: 140.8s

Plot:



After we plug the points into the curve fitting tools, I get this curve and whose formula almost same as y= x^2.