Assignment 1 – Nearest Neighbor

[Carlos Gomez]

April 28, 2017

1 Problem Description

- Take a set of points in a plane, $\{p_1 = (x_1, y_1), p_2 = (x_2, y_2), ..., p_n = (x_n, y_n)\}$ and find the distance between the closest pair of points using the formula:

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- For this problem there will be two methods that will be used to solve this problem. The methods will be the Brute Force method and the Divide-and-Conquer method.

- The Brute Force method will be good to use to compare my Divide-and-Conquer method with. The Divide-and-Conquer method will be used to show how fast we can solve this problem if we break it down into easier problems.

2 Brute Force

- The Brute Force method that was used in finding the minimum distance between the closest pair of points was just to compare half of all the points in the set with all the points in the same set of points.

- The run time of this algorithm is $O(n^2)$, it has a quadratic run-time.

- The main purpose of using a Brute Force method was for comparison purposes. It allowed me to check if I was obtaining the correct answer for my divide-and-conquer algorithm.

3 Divide-and-Conquer

- The Divide-and-Conquer method is a recursive method that divides the problem up into smaller problems and makes the smaller problems easier to solve. In this problem all of our points are in a list of tuples, we break down this list in half and recursively call our divide and conquer function for each halves. These recursive calls return the minimum distance in each half. We then have to compare the middle portion, due to the possibility of some points having a smaller distance that is between the left and right portion.

- In each iteration of the Divide-and-Conquer function we call the function twice and divide the problem up into two smaller portions. Once returned from the recursive function, we compare both minimum values from either side and keep the smaller distance, δ. We combine all the points from both sides that fulfill the requirements $x_i > x_{mid} - \delta$ and $x_i < x_{mid} + \delta$. The for every point in this list we compare it to the next 7 points.

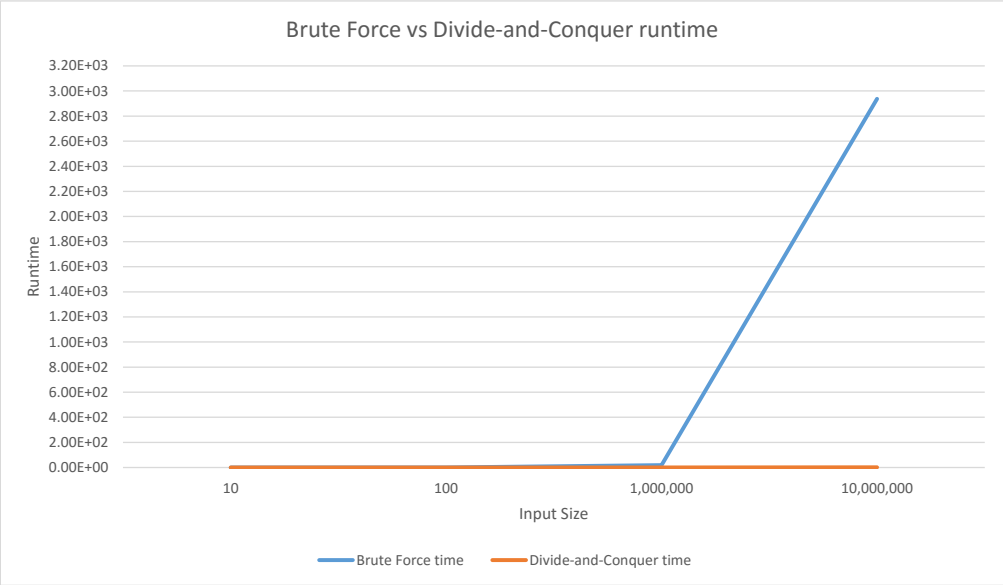
- It's better than the Brute Force method if we have more than 10 items in our list of points. Because of the nature of our algorithm by checking the next 7 points for each point in the middle section, if we get below 10 points, the Divide-and-Conquer method run-time is about $\Theta(n^2)$. If we got above 10 points the Divide-and-Conquer algorithm is much faster than the Brute Force method. Instead of checking all the points to each point, we just check the next 7 points and see if we can find a distance that is smaller than our previously acquired distance.

- We obtain the recursive formula $T(n) = 2T(\frac{n}{2}) + 7n$. Using Master's Theorem we find out this is case 2 of Master's Theorem, so $\Theta(n^{log_2 2} log^1 n)$.

- The run-time for this algorithm is $\Theta(nlogn)$

4 Results

	Brute Force		Divide-and-Conquer		
Benchmarks	$O(.)$	Actual	$O(.)$	Actual	Minimum Distance
input10.txt	$O(n^2)$	4.35e-05	O(nlogn)	9.79e-05	2.8635642126552705
input100.txt	$O(n^2)$	0.00178	O(nlogn)	0.000449	9.91036114377263
input10e5.txt	$O(n^2)$	18.94	O(nlogn)	0.118	0.07858753081757049
input10e6.txt	$O(n^2)$	2937.43	O(nlogn)	1.61	0.01664331697710184



5 Conclusions

- As we can see from the results we can see that if we have a list size of 10, the Brute Force method is faster than the Divide-and-Conquer method. Above a list size of 10, the Divide-and-Conquer method starts to become considerably faster than our Brute Force method.