Caleb Wagner (cwagner)
CS2223 Algorithms Project 2
November 21, 2017

Closest-Pair Problem Analysis

The purpose of this project is to compare the time efficiencies for two algorithms used to calculate the minimum distance between a set of points. The first algorithm uses a brute force approach that checks the distance of every point in a given list of coordinates and returns the smallest distance. The second algorithm uses a recursive divide and conquer method to find the smallest distance. This algorithm divides a given (sorted) list of coordinates in half and calls itself recursively on both halves. A time experiment was performed for both algorithms as well as a theoretical analysis of the two algorithms (which can be found on page 3 and 4 of this report).

The time experiment was conducted by saving the CPU time before each algorithm was run and then saving the time after each algorithm was run. The difference between the start and end times was considered the run time of the algorithm. This process was repeated twenty times for each algorithm and the average time calculated in order to ensure an accurate result. Based on the results from the various test cases in addition to the analysis of the structures of the algorithms, the time efficiencies for the algorithms can be described as follows:

Table 1: Time Efficiencies of Algorithms		
	Brute Force Algorithm	**Recursive Algorithm**
Big O Notation	n^2	n(log(n))

Table 1: Time efficiencies for the two algorithms

In addition to comparing the time efficiencies of the two algorithms experimentally, efficiency equations were created and solved to determine a worst-case efficiency. Please see the last two pages of this report to see the equations that were generated as well as how they were simplified/solved. Solving these equations gave values of $O(n^2)$ for the brute force algorithm and O(n(log(n)) for the recursive algorithms. These answers make sense based upon an analysis of the implementation of each algorithm. The brute force algorithm uses a nested for loop. Both the outer for loop and the nested for loop range from 0 to n and (i+1) to n. As such, these nested for loops would cause an efficiency of n^2. The recursive algorithm is a little harder to analyze just by looking at code, but by comparing the theoretical and experimental values, O(n(log(n)) for the recursive algorithm seems a valid worst-case notation.

It can therefore be concluded that for larger inputs, the recursive algorithm is more efficient than the brute force. However, for small values, the brute force is faster than the recursive. As currently implemented where the recursive algorithm is written based on the given pseudocode, if a list of size three or less is given, the recursive algorithm uses brute force to find the minimum distance. As an interesting result of this experiment, it was found through experimentation that this number could be raised to something higher, such as a list of size five.

Calling the recursive algorithm on a small list is not always the most efficient. As such, the brute force is more efficient for smaller input and the recursive is more efficient for larger input.

Table 2: Experimentation of Different Values for the Two Algorithms				
Input Points	**Brute Force Algorithm Runtime (s)**	**Recursive Algorithm Runtime (s)**	**Shortest Distance**	**Comments**
[(0,1), (99,50), (153,22), (44,11)]	0.0000175	0.0000145	45.12205669	Recursive Algorithm is the fastest, although there is a tendency for the algorithm to oscillate between recursive and brute force (caused by small number of inputs).
[(3,666), (616,223), (27,54), (69,73), (157, 323), (415,179)]	0.0000374	0.0000277	46.09772228	Recursive algorithm is the fastest, large number of inputs chosen.
[(0,0), (7,6), (2,20), (12,5), (16,16), (5,8), (19,7), (14,22), (8,19), (7,29), (10, 11), (1,13)]	0.0001503	0.0000634	2.8284271247	Recursive algorithm is the fastest, large number of inputs chosen.
[(0,0), (-4, 1), (-7,-2), (4,5), (13,6)]	0.0000428	0.0000368	4.242640687	Brute force algorithm is the fastest, small number of inputs chosen.
[(2, 3), (12, 30), (40, 50), (5, 1), (12, 10), (3, 4)]	0.0000365	0.0000260	1.4142135623	Recursive algorithm is the fastest.
[(0, 1), (1, 2), (3, 3)]	0.0000146	0.0000184	1.4142135623	Brute force is the fastest, small number of inputs chosen

Table 2: Experimentation results for the algorithms

Brute Force Efficiency Calculation

This equation was created by analyzing code within the nested for loop. The math performed within in the nested for loop is constant time. The nested for loop ranges from k = i + 1 to n-1 (the size of the input list), and the outer loop ranges from 0 to n-2. As such, an equation was created describing the efficiency as:

$$T(n) = \sum_{i=0}^{n-2} 1 \sum_{k=i+1}^{n-1} 1 \qquad \text{Equation 1}$$

Expanding the first summation gives:

$$T(n) = \sum_{i=0}^{n-2} ((n-1) - (i+1) - 1) \qquad \text{Equation 2}$$

This result can be simplified as follows:

$$T(n) = \sum_{i=0}^{n-2} (n - i - 3) \qquad \text{Equation 3}$$

The result can be broken into three separate summations:

$$T(n) = \sum_{i=0}^{n-2} n - \sum_{i=0}^{n-2} i - 3\sum_{i=0}^{n-2} 1 \qquad \text{Equation 4}$$

Solving the summation gives:

$$T(n) = n(n-1) - \frac{n(n-1)}{2} - 3(n-2) \qquad \text{Equation 5}$$

Which, after dropping constants and smaller terms, can be simplified to:

$$T(n) = O(n^2) \qquad \text{Equation 6}$$

Recursive Algorithm Efficiency Calculation
The recursive equation for the recursive algorithm can be written as:

$$T(n) = 2T\left(\frac{n}{2}\right) + f(n)$$ Equation 7

2T(n/2) describes calling the algorithm two times recursively as well as each time the recursion is called, the size of the input list is divided in half. f(n), which is the code not part of the recursive element of the algorithm, is used to check points that are directly half from the middle of the coordinate frame (lie on the middle vertical line of the set of points). All of these points need to only be checked once for the minimum distance. As such, this creates a linear function, which in turn makes f(n) linear.

Using the Master Method, this equation can be compared as:

$$n^{\log_2 2} \; vs \; n$$ Equation 8

This result can be simplified as follows:

$$n \; vs \; n$$ Equation 9

As per the Master Method, since the two comparisons are equal, the final result is given by:

$$T(n) = O(n \log n)$$ Equation 10