CS 310
Assignment 321

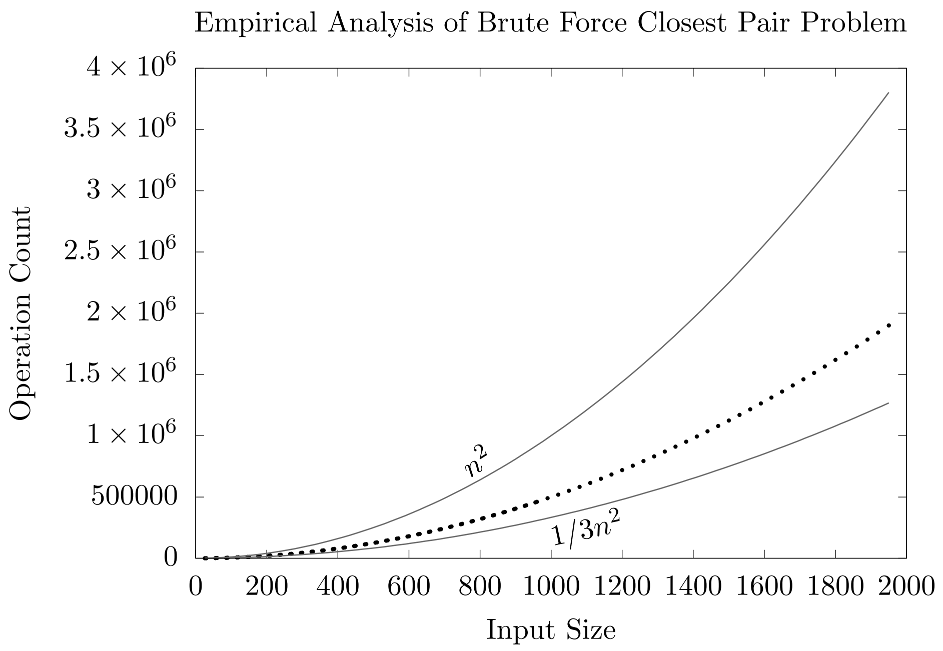
Cameron Moberg

There are several ways to solve the closest-pair problem, the two done in this analysis are the brute-force method, where you compare every single pair and find the smallest, and the divide-and-conquer method. The purpose of this study is to perform an empirical analysis of the two methods.

For both methods, the value for n is taken to be the number of points in the plane.

To analyze the brute-force algorithm, we chose calculating the distance on line 67 as the basic operation, because this happens most frequently, is the most expensive, and is the philosophical heart of this algorithm.

An empirical analysis of running the algorithm for multiple values of n produces the results shown below. Standard function $f(n) = n^2$ with constant multipliers, has been added to illustrate the analysis.



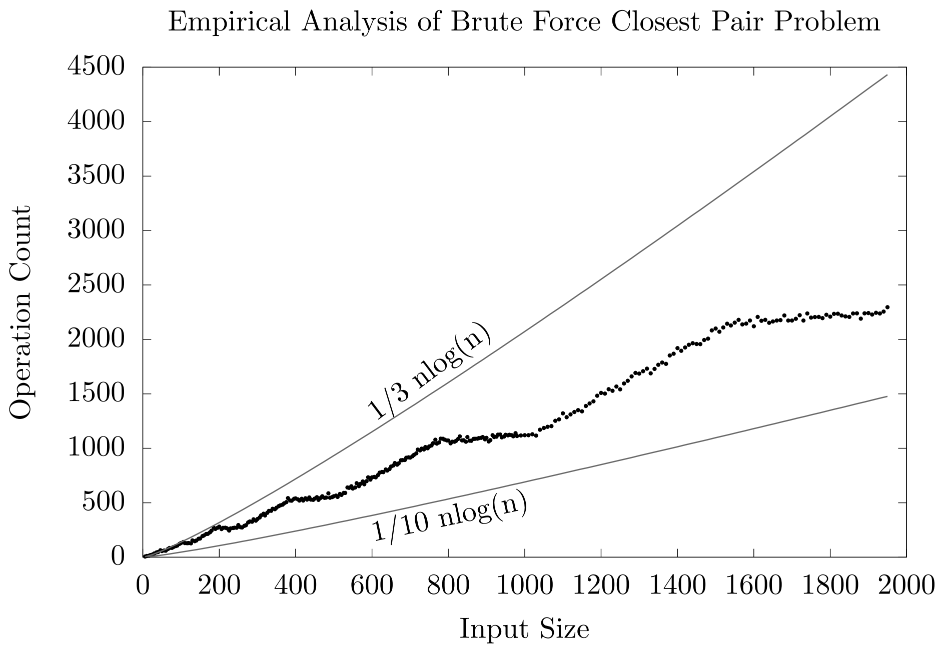
An examination of the code itself explains the empirical results when we observe that there is a doubly-nested for loop that iterates over the input size.

Therefore, since the brute-force algorithm always runs to completion we conclude that the algorithm is described by

$$T(n) \in \Theta(n^2)$$

To analyze the divide-and-conquer algorithm, we chose calling the distance method on lines 92 and 99–103 and calculating the distance on lines 150–153 as the basic operations, because these happen most frequently, and are the philosophical heart of this algorithm. There was also the option of choosing the comparison to find the minimum distance (lines 102–103), but compared to finding the distance, it is much less expensive to compare two numbers rather than compute.

An empirical analysis of running the algorithm for multiple values of n produces the results shown below. The standard function $f(n) = n \log(n)$ with constant multipliers, has been added to illustrate the analysis.



An examination of the code itself explains the empirical results when we observe that this algorithm is recursive, lending us the ability to use the Master Theorem, which is of the form

$$T(n) = aT(\tfrac{n}{b}) + f(n) \text{ where } a \geq 1, b > 1$$

Now, this algorithm has two recursive calls, each recursive call operates on half of the input, and $f(n) \in \Theta(n)$ where $f(n)$ is the local work, because both dividing the problem in half and combining the solutions take linear time. Thus, applying the Master Theorem with a, b, and d equal to $2, 2$, and 1 respectively, we get $T(n) \in \Theta(n \log(n))$.

In addition, the input does not need to be pre-sorted, since using a $O(n \log(n))$ efficiency sorting algorithm will not change the asymptotic behavior of this algorithm.

Therefore, since the divide-and-conquer algorithm always runs to completion we conclude that the algorithm is described by

$$T(n) \in \Theta(n \log(n))$$