Kamil Gumienny Project 2
CS2223- Closest Pair

This project compared two algorithms that compute the smallest distance between two points on a Cartesian plane.

- The first algorithm is non-recursive and checks every point against itself using the distance formula $\sqrt{\left(x_i - x_j\right)^2 + (y_i - y_j)^2}$.
    - This will yield the correct answer without relying on memory but the run time would be O(n^2) because every point is being computed against every other point.

- The second algorithm is a recursive algorithm that calls the distance formula to compute distance but will only find the distance between a max of 3 points that are nearby at a time, so the work on each recursion call will not reach O(n^2).
    - The algorithm sorts the points by their x coordinates and their y coordinates.
        - This is only done once and can take as little as O(nlogn) time using the algorithms we learned, so it does not get counted into the recursive algorithm.
    - It then uses those sorted lists to divide the points into two separate planes where it keeps dividing based on the points x and y coordinates until it computes the smallest distance.
        - This takes O(n) time. Every point is multiplied with 2 other points that are near it.
    - Once the smallest distance is computed in the subsets of the original plane, the algorithms takes the smallest distance and compares points along the original splitting point (first recursive call) to see if any points from the two separate halves have closer distances than the points within their subsets. Lastly it returns the smallest distance.
        - It takes O(n) time to divide arrays P and Q into Pl, Pr, Qr, and Ql and O(n) time to check the points along the original splitting point. Both these steps have a basic operation in a single for-loop.

After completing test cases. It appears if the input is small enough, the brute force method is quicker because you avoid adding and removing from the stack and all the recursion calls that take time. However, with a large enough input, the time difference does start making a difference.

Solving for Recursive Algorithm

T(n)= 2T(n/2)+O(n) #Two recursive calls with half the original arrays in them. See above for O(n)
a=2, b=2, f(n) = n^1
$n^{\log_2 2} == n^1$ #Case 2 of Master Method
T(n)=T(nlogn)

Test Cases

Input	Closest Distance	Brute force time	Recursive time
[(0,1),(99,50),(153,22),(44,11), (0,10)]	9	24.9 µs	43.72 µs
[(0, 0),(7, 6),(2, 20),(12, 5),(16, 16), (5, 8),(19, 7),(14, 22),(8, 19),(7,29), (10, 11),(1, 13)]	2.8284	24.59 µs	62.06 µs
[(0, 434),(754, 634),(232, 204),(122, 435),(166,186),(435,868),(194,723),(154, 222),(878,192),(732,249),(310,141),(781, 137),(560,805),(754,675),(452,207),(152, 745),(716,156),(576,845),(139,567),(154, 422),(854,159),(227,294),(120,143),(531, 143),(430,540),(327,645),(322,240),(152, 521),(136,165),(425,853),(419,754),(124, 242),(821,194),(723,249),(150,411),(231, 134),(230,304),(457,653),(532,206),(124, 523),(146,156),(255,865),(169,743),(154, 262),(845,169),(743,529),(103,141),(153, 173)]	9	1569.7 µs	551.9 µs