Closest Pair of Points Project

Daniel Selvidge, Mia Guzman

Problem Overview and Methods

The Closest Pair Problem is a computational geometry problem, where given n points in metric space, find a pair of points with the smallest distance between them. In this experiment, both the divide-and-conquer and KD\Ball Tree method will be used. The divide-and-conquer method uses array representation to divide the array in mostly half, copies both halves of the array, then recursively calls itself to each half to find the pair of closest points. KD\Ball Trees use a binary tree representation. KD Trees work better with lower dimensional datasets, splitting its data using hyperplanes. After this division, a binary tree is created where it will be traversed to find the closest pair of points. Ball Trees work very similarly, working better with higher dimensional datasets, they split their data using hyperspheres instead. Though both have their preferred dataset dimensions, both work best with larger datasets which reduces time complexity.

Research Questions

The questions that we seek to answer through this experiment are as follows:

At what input size does divide-and-conquer outperform brute-force?

Do KD Trees and Ball Trees show measurable performance differences in 2D versus higher dimensions?

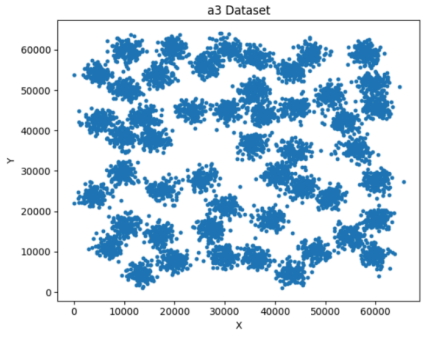
How do point distribution and data size affect each approach?

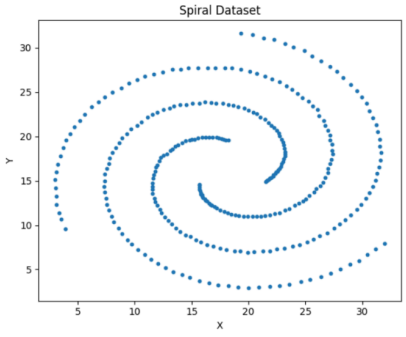
Experiment Description

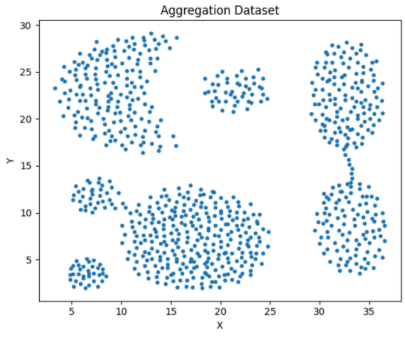
The project was implemented using Python in a Jupyter Notebook. The datasets used were a general, spiral, and an aggregation of clusters. These datasets were retrieved from Kaggle as they were clean datasets that could be used for this project. Another dataset is used which is made up of random points that start at size 4 then exponentially grow to 256 points. The reference method used is the brute-force method of the closest pair problem. The divide-and-conquer method and the KD Tree method then have their execution times measured and compared. This comparison occurs with the randomly generated dataset and the three previously mentioned datasets. A graph is also shown to visualize the execution times between the two methods.

Results

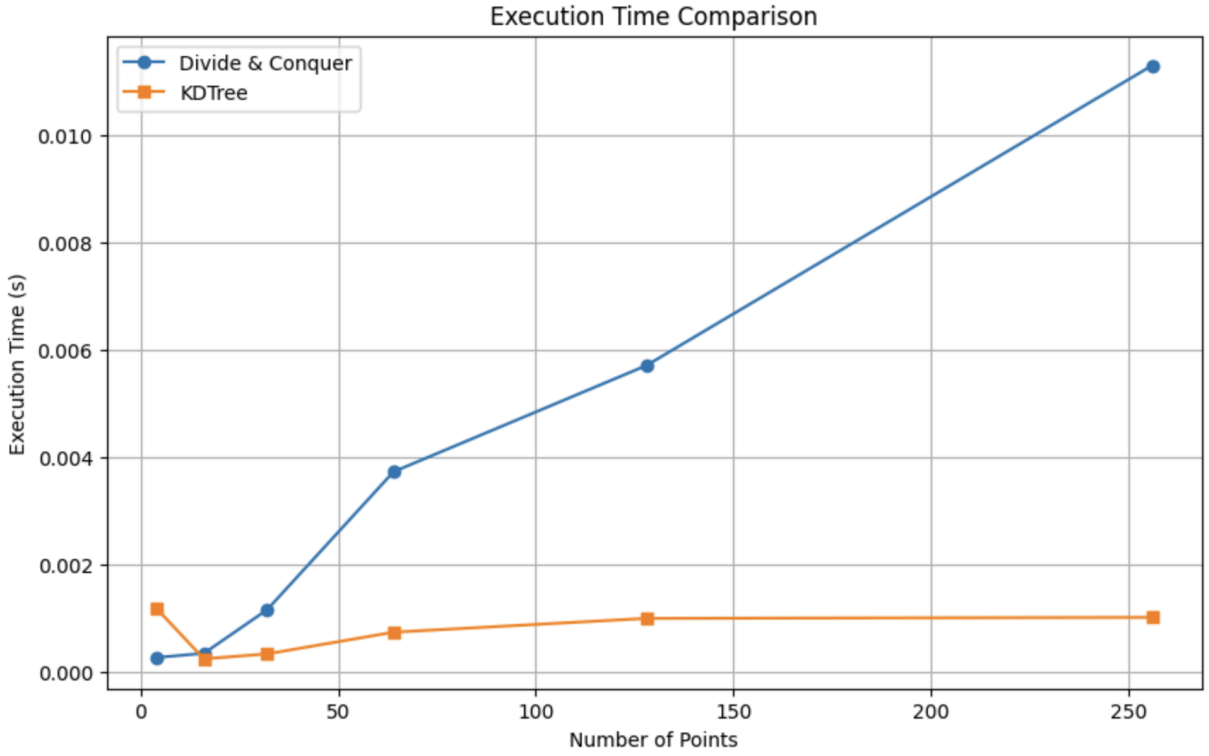
Our results show that the brute-force method, while conceptually simple, is impractical for large datasets due to its quadratic time complexity. In contrast, the divide and conquer algorithm provided a substantial improvement in runtime, aligning with its expected O(n log n) performance. The KD-Tree approach consistently performed the fastest in our empirical analysis, particularly on larger or denser datasets, though its effectiveness can vary based on data distribution. One limitation of our experiment is that we only explored three methods despite outlining others, such as randomized algorithms or Delaunay triangulation, which could offer additional insights. We also only tested on two-dimensional datasets and did not average runtime across multiple trials, which could improve accuracy. If we repeated the experiment, we would include more algorithms from our original outline, run multiple iterations for each test, and explore scalability in higher dimensions. These findings not only validate the theoretical performance of the algorithms but also raise further questions about performance trade offs in high dimensional or real world spatial data, suggesting several promising directions for future research.







	Dataset	DC Distance	DC Time	KD Distance	KD Time
0	a3	3.162278	0.185013	3.162278	0.005896
1	spiral	0.070711	0.003967	0.070711	0.000282
2	aggregation	0.111803	0.016816	0.111803	0.000495



	n	Divide&Conquer Time	KDTree Time
0	4	0.000263	0.001178
1	16	0.000341	0.000238
2	32	0.001153	0.000330
3	64	0.003726	0.000735
4	128	0.005707	0.000992
5	256	0.011299	0.001011

Percentage	Mia	Daniel
Brainstorming	50%	50%
Coding	0%	100%
Experiment Design	50%	50%
Visualization / Results	10%	90%
Communication	50%	50%
Submission	90%	10%
Report Writing	80%	20%
Presention	100%	0%

Difficulties and Roadblocks

One of the main challenges we faced during this project was coordinating our schedules. With both of us having busy course/work loads and other commitments, it was often difficult to find time to sit down and work together consistently. This led to delays in testing different algorithms and finalizing our analysis. Another roadblock was related to hardware limitations since we were both working on standard laptops, we occasionally ran into slow processing times when testing with the datasets, especially with the brute-force approach. While these issues didn't prevent us from completing the project, they did add extra time and effort to our workflow and made it harder to experiment as much as we initially planned.

Sources

Brainstorming

"Closest Pair of Points Problem." *Wikipedia*, Wikimedia Foundation, 29 Dec. 2024, en.wikipedia.org/wiki/Closest_pair_of_points_problem.

"How to Reduce KNN Computation Time Using KD-Tree or Ball Tree?" *GeeksforGeeks*, GeeksforGeeks, 23 July 2025, www.geeksforgeeks.org/machine-learning/how-to-reduce-knn-computation-time-using-kd-tree-or-ball-tree/.

"Ball Tree and KD Tree Algorithms." *GeeksforGeeks*, GeeksforGeeks, 23 July 2025, www.geeksforgeeks.org/machine-learning/ball-tree-and-kd-tree-algorithms/.

1 Preliminaries 2 o(Nlog N) Divide and Conquer Algorithm, www.cs.cmu.edu/~15451-s20/lectures/lec23-closest-pair.pdf. Accessed 29 July 2025.

"Closest Pair of Points: O(NLOGN) Implementation." *GeeksforGeeks*, GeeksforGeeks, 23 July 2025, www.geeksforgeeks.org/dsa/closest-pair-of-points-onlogn-implementation/.

Wu, Shih-Yu (Fiona). "Algorithms Studynote-4: Divide and Conquer - Closest Pair." *Medium*, Medium, 5 Feb. 2025, medium.com/@shihyu-wu/algorithms-studynote-4-divide-and-conquer-closest-pair-49ba679ce3c7.

Implementation

Comment, et al. "Minimum Distance between Two Points." *GeeksforGeeks*, 23 July 2025, www.geeksforgeeks.org/dsa/closest-pair-of-points-using-divide-and-conquer-algorithm/.

"KDTree." *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.neighbors.KDTree.html. Accessed 28 July 2025.

Zybook Chapters 14.1 (Divide and Conquer) and 24.4 (KD Algorithm)
Chatgpt: Used in enhancing writing, troubleshooting code