Closest Pair Problem

Divide-and-Conquer & KD/Ball Trees

Daniel Selvidge, Mia Guzman

Problem Overview & Methods

- Computational geometry question
- Given n points in metric space, find a pair of points with the smallest distance between them

- **Divide-and-Conquer:** Using array representation, divide array into mostly half, then copy both sides and recursively search for the closest point

- **KD/Ball Trees:** Using binary tree representation, KD trees split the data using hyperplanes, then create a binary tree which is traversed to find the closest pair. Ball trees work similarly, using hyperspheres to split the data

Research Questions

- At what input size does divide-and-conquer outperform brute force?
- Do KD Trees and Ball Trees show measurable performance differences in 2D vs. higher dimensions?
- How do point distribution and data size affect each approach?

Experiment Description

- Implementation: Python, Jupyter Notebook
- Input: clean datasets used from Kaggle (general, spiral, aggregation of clusters)
- Input: Randomly generated points ranging from 4 – 256
- Reference method: Brute-force closest pair
- Comparisons of execution times
- Graph of execution times

Results

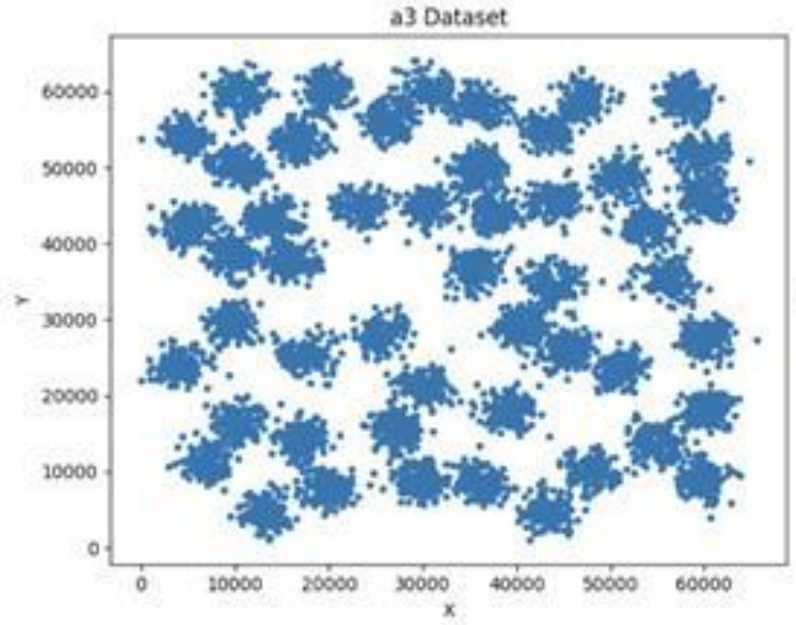
- Brute-force method is impractical for larger datasets
- Divide-and-Conquer provides substantial improvement in runtime
- KD Tree consistently performed the fastest
- Three methods were explored, despite outlining others
- Only used 2D datasets
- Runtime was not averaged across multiple trials

Results: What to do differently

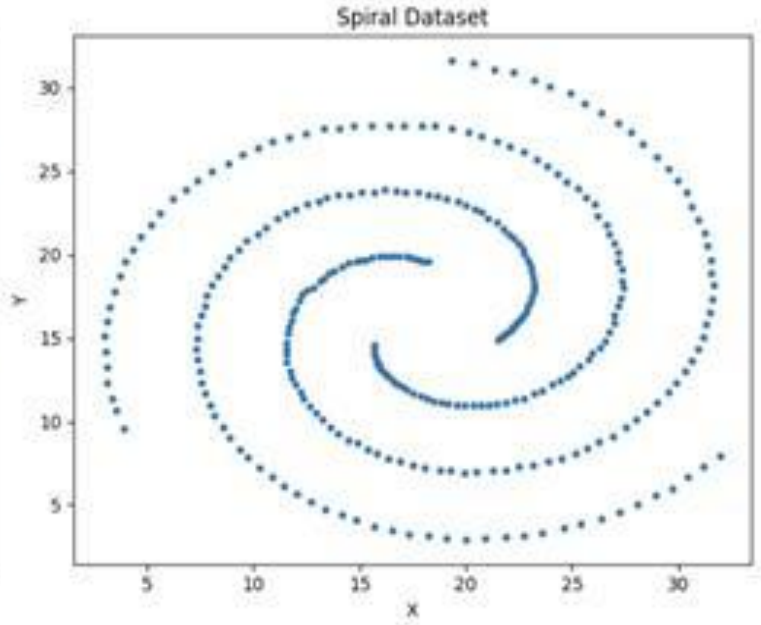
- Include more algorithms from original outline
- Run multiple iterations for each test
- Explore scalability in higher dimensions

Dataset Visualization

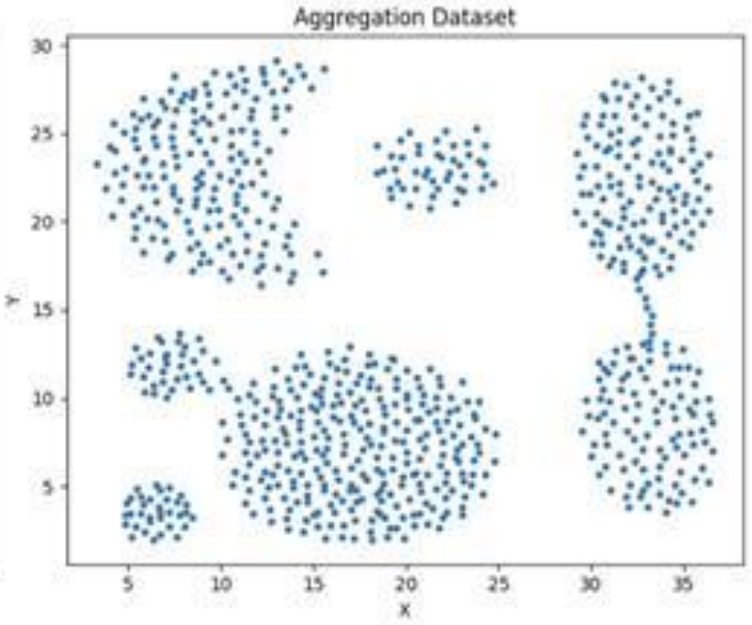
a3: Clustered Blobs



Spiral: Smooth Curve Structure



Aggregation: Tight regional Groupings



Distance Results

- Both methods found identical closest distance
- Confirms correctness of KD-Tree and Divide and Conquer implementations
- Consistency across datasets

	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

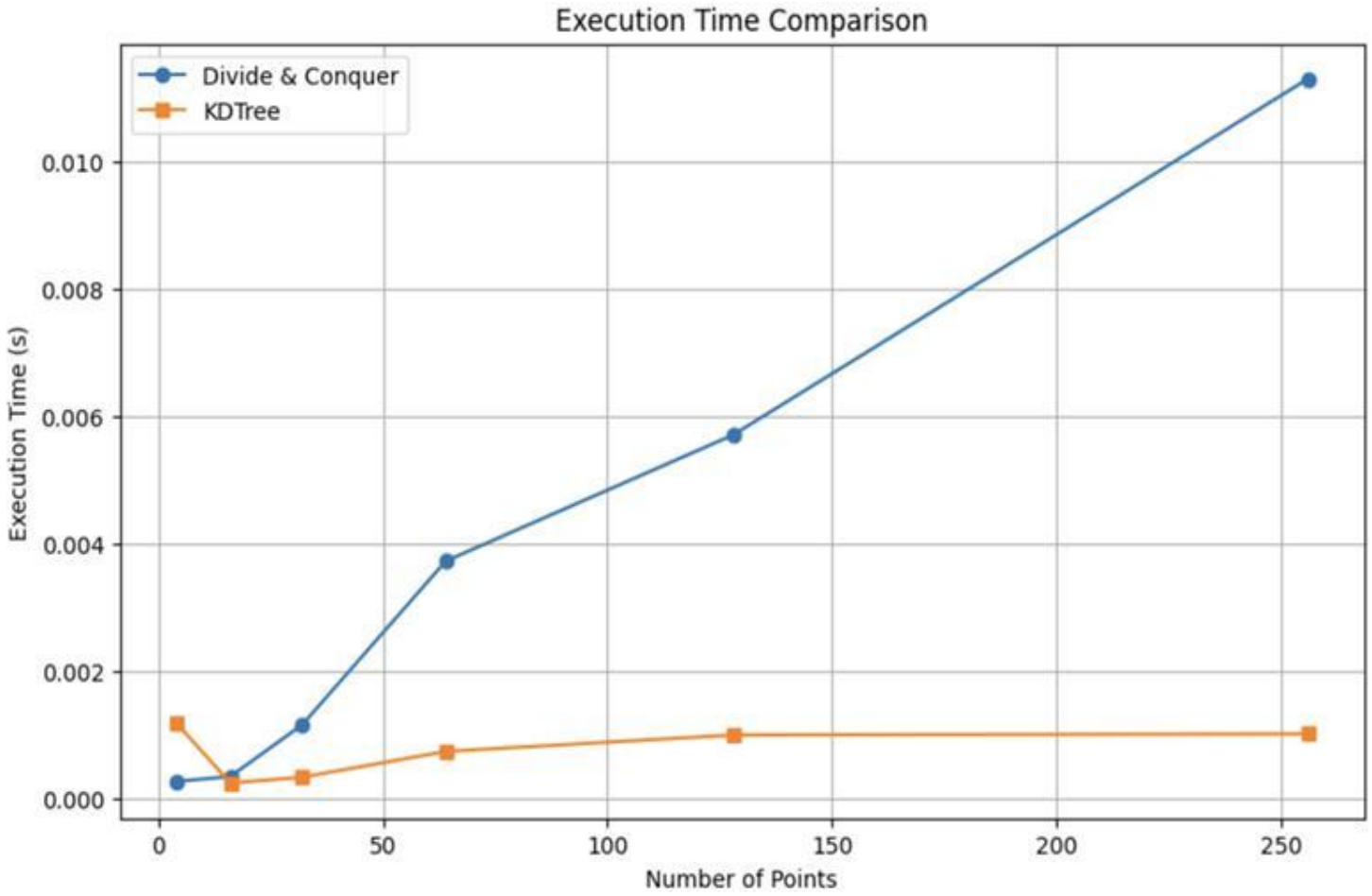
Execution Time by Dataset

- KD-Tree completed in under 0.001s in most cases
- Divide and Conquer was 3-10x slower, but still scalable
- Big gap on the a3 dataset

	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

Time vs Dataset

- Divide and Conquer grows with O(n log n) pattern
- KD-Tree runtime nearly flat which shows extreme efficiency
- Great choice for large scale spatial data



Difficulties and Roadblocks

- Coordinating schedules
  - Jobs, class work, personal life
- Hardware limitations
  - Working on laptops

Sources

- “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/.

Sources

- 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