Lab Report 2

Convex Hull

by **Kefaet Ullah (2103011)**

to **A. F. M. Minhazur Rahman (Assistant Professor, CSE, RUET)**

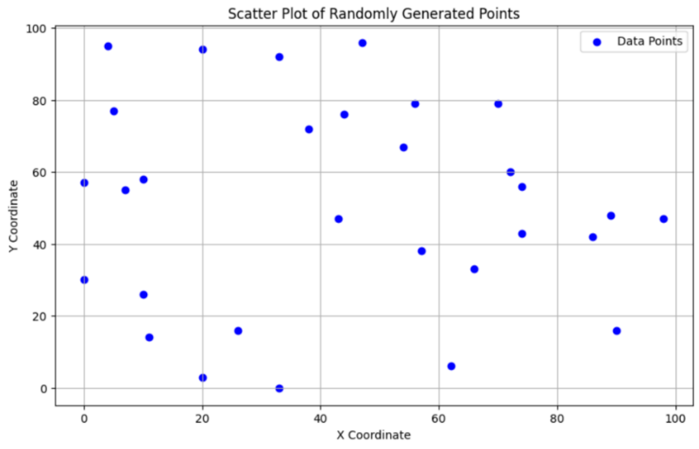
Task 1

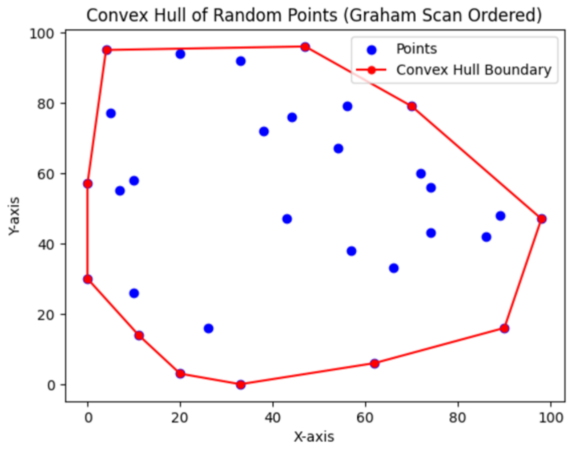
Problem Statement: Implement and compare the Quickhull and Graham Scan algorithms to compute the convex hull of 2D points, visualize the results, and analyze execution times on randomly generated datasets of varying sizes.

Code:

`Quick Hull Algorithm:` https://colab.research.google.com/drive/1d-bBknRfz0CJseO150pPCXJ3WJ90-zvT?usp=sharing
`Graham Scan Algorithm:` https://colab.research.google.com/drive/1R-D1vOFAO_ydS0q6BVFiqJE0TvaMN_I?usp=sharing

Output:

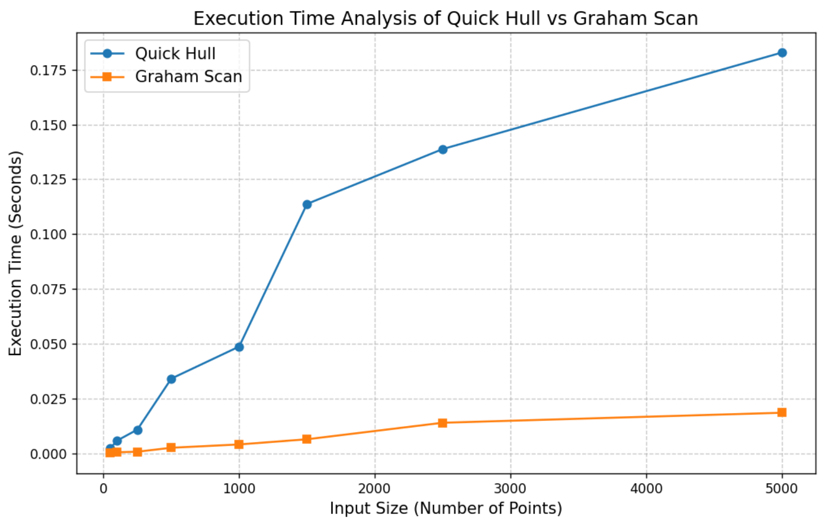




Comparison:

Size	Quick Hull (s)	Graham Scan (s)
50	0.002534	0.000347
100	0.005983	0.000686
250	0.011022	0.000934
500	0.034236	0.002739
1000	0.048835	0.004252
1500	0.113808	0.006608
2500	0.138793	0.014129
5000	0.182765	0.018683

Result Analysis:



Overview of the Algorithms:

- Quick Hull Algorithm: Quick Hull is a divide-and-conquer algorithm that computes the convex hull by recursively finding the farthest points from a dividing line and constructing hulls for subproblems. It operates similarly to QuickSort in partitioning the problem.
- Graham Scan Algorithm: Graham Scan uses sorting and a stack-based approach. It begins by selecting the point with the lowest y-coordinate (and lowest x in case of ties), sorts the remaining points by polar angle, and constructs the convex hull by traversing the sorted list while maintaining the convexity condition.

Observation from the Graph: The execution time of Quick Hull algorithm increases more sharply with the input size compared to Graham Scan, especially for larger datasets. Grahma Scan algorithm demonstrates consistently better performance due to its more predictable and efficient operations, particularly for large datasets.

Theoretical Analysis:

- Quick Hull:
  Steps: a)Finding the farthest point from a line: *O(n)*. *b)*Dividing the points into two subsets relative to this line: *O(n)*. *c)*Recursively solving each subset.
  Recurrence Relation: Let *T(n)* be the time complexity for n points. *T(n)=2T(n/2)+O(n)* This is similar to QuickSort's recurrence.
  Using the Master Theorem: *T(n)=O(n logn)* in the *average case*.
  In worst Case, T(n)=T(n−1)+O(n)
  Solving this: *T(n)=O(n²)*
- Graham Scan:
  Steps: a) Finding the starting point (lowest y-coordinate): *O(n)*. b) Sorting all points by their polar angle relative to the starting point: *O(n logn)*. c) Constructing the convex hull by traversing the sorted points and using a stack: *O(n)*.
  The sorting step dominates, so the overall time complexity is *O(n logn)*.

$$\text{Ratio} = \frac{Quick\ Hull\ Worst\ Case}{Graham\ Scan} = \frac{n^2}{n\ logn} = \mathrm{O}\left(\frac{\mathrm{n}}{\mathrm{logn}}\right)$$

This shows that as n increases, Quick Hull's worst-case time grows significantly faster than Graham Scan.

Discussion & Conclusion: Graham Scan has a predictable time complexity of *O(n logn)*, making it robust for practical use. Quick Hull's divide-and-conquer approach is efficient on average, but its *O*(n^2) worst-case complexity makes it unsuitable for large datasets with unfavorable distributions. For real-world scenarios with large and complex datasets, Graham Scan is the preferred choice due to its consistent performance. Quick Hull can still be useful for smaller datasets or cases where input distribution favors divide-and-conquer strategies.